

USER GUIDE

Essential Studio for File Formats

Version - v18.1.0.36 | Release Date - March 19, 2020

Welcome to Syncfusion Essential File Formats Platform.....	30
How to best read this user guide	30
Additional help resources	30
Create a support incident	30
List of File Format libraries:	30
Quick Start links:	31
System Requirements	31
Operating Systems	31
Hardware Environment.....	31
Development Environment.....	31
Download Installer	31
Download the Trial Version	31
Free Trial Page.....	31
Start Trial Page	32
Download the License Version.....	33
Installation and Upgrade.....	33
Installation using Web Installer	33
Overview	33
Installation	34
Uninstallation.....	44
Installation and Uninstallation	49
Installation using Offline Installer	58
Installing with UI	58
Installing in silent mode	65
Installation FAQ.....	67
Upgrade from one version to another version.	67
Upgrade from major version to service pack version	67
Upgrade from trial version to license version.....	67
NuGet Packages	67
Installing NuGet Packages.....	67
Using NuGet Package Manager	67
Using Package Manager Console	69
DocIO.....	69
Creating a new Word document with few lines of code	69
Creating a new Word document from scratch with basic elements	71

Modifying an existing Word document	84
Performing Mail merge	87
Converting Word document to PDF.....	99
Document Object Model representation	102
Loading & saving document.....	104
Opening an existing document	104
Opening an existing document from Stream.....	105
Opening an Encrypted Word document	107
Opening the read only Word document.....	109
Saving a Word document to file system	110
Saving a Word document to Stream	112
Sending to a client browser	114
Closing a document.....	115
Working with Word document	117
Iterating through document elements	117
Cloning a Word document	134
Merging Word documents.....	138
Printing a Word document	146
Working with Styles	150
Working with Word document properties	157
Setting the Background for a Word document.....	162
Working with Sections	167
Specifying Page Properties.....	172
Creating Multi-column document.....	174
Creating document with different page settings.....	178
Working with Headers and Footers	181
Adding Page Numbers.....	197
Removing a Section.....	207
Working with Paragraph	208
Applying paragraph formatting.....	213
Working with text	226
Working with Images	234
Working with lists	249
Working with hyperlinks	277
Working with symbols.....	291

Appending breaks	296
Appending OLE objects	299
Working with Text Box.....	302
Working with Tables	308
Nested Table	321
Align text within a table	325
Apply formatting to Table, Row and Cell	326
Merging cells vertically and horizontally	347
Specifying table header row to repeat on each page	356
Keeping rows from breaking across pages	359
Iterating through table elements.....	360
Working with Bookmarks.....	363
Adding a bookmark	363
Obtaining a bookmark instance	366
Removing a Bookmark from Word document.....	368
Retrieving contents within a bookmark.....	370
Inserting content into a bookmark	375
Deleting content from a bookmark	387
Replacing content in a bookmark	389
Working with document Fields	397
Adding fields.....	397
Formatting fields	399
Updating fields	402
IF field.....	404
Document variables	407
Cross reference	410
Unlink fields.....	413
Working with Shapes	415
Adding shapes	415
Grouping shapes	425
Ungrouping shapes	451
Working with Mail merge	453
Mail merge process.....	454
Simple Mail merge	459
Performing Mail merge for a group	459

Performing Nested Mail merge for group	459
Performing Mail merge with dynamic objects.....	459
Performing Mail merge with business objects	459
Performing Nested Mail merge with relational data objects	459
Event support for mail merge	459
Mail merge options	460
Find and Replace.....	460
Finding contents in a Word document	460
Replacing the Search results	468
Working with Table Of Contents.....	476
Adding a TOC field.....	476
Updating table of contents	478
Creating table of contents with user-defined styles.....	479
Applying Watermark	482
Text Watermark	482
Picture Watermark.....	485
Working with Comments	486
Adding a Comment	486
Modifying a Comment	488
Removing Comments.....	490
Working with Form Fields	493
Check Box.....	493
Drop-Down.....	499
Text Form field	504
Working with Charts	511
Creating a simple Chart from scratch	511
Creating a Chart from Excel file	517
Creating Custom Charts	520
Refreshing the Chart	525
Modifying the Chart data.....	527
Customizing the Chart & its elements	529
Applying 3D Formats.....	549
Removing Chart.....	551
Supported Chart Types	553
Security	555

Encrypting with password.....	555
Opening the encrypted Word document.....	556
Remove encryption.....	558
Protecting Word document from editing	559
Footnotes and endnotes.....	561
Adding a Footnotes	561
Adding a Endnotes	564
Footnote and Endnote separators	568
Working with Macros.....	575
Working with Content Controls	579
What is Content Control?	579
Common properties of Content Control.....	584
Why Content Control?	589
Types of Content Controls	633
Working with Mathematical Equation.....	652
Types of equation	652
Modify existing equation	718
Accepting or Rejecting Track Changes	724
Conversion	728
Working with Document Conversions	728
Converting Word document to PDF.....	729
Rendering / Converting Word document to Image	729
RTF conversion.....	729
HTML conversion	729
Text file.....	730
Word to EPUB	730
Supported and Unsupported Features	730
Word document features.....	730
Blazor supported features	737
FAQ/How to	739
How to modify an existing style?	739
How to open a document from stream using DocIO?	740
How to set OpenType Font Features?	742
How to attach a Template to a Word document?	747
How to insert a DataTable in a Word document?	747

How to insert a table from HTML string in Word document?	750
How to set table cell width?	750
How to position a table in a Word document?.....	751
How to set the text direction to a table in Word document?	752
How to extract the images in the document?	753
How to remove headers and footers from the document?	754
Which units does Essential DocIO uses for measurement properties such as size, margins, etc, in a Word document?	756
Could not find Syncfusion.OfficeChartToImageConverter assembly in .NET 3.5 Framework, does it mean there is no support for chart conversion in this Framework?	756
Can the chart data be refreshed?	756
Is it possible to convert 3D charts to PDF or image?	756
Is it possible to specify PDF conformance level in Word to PDF conversion?	756
Migration from Microsoft Office Automation to Essential DocIO	757
How to copy necessary fonts to Linux containers	780
PDF	780
Creating a PDF document with simple text	781
Creating a PDF document with image	783
Creating a PDF document with table	785
Creating a simple PDF document with basic elements.....	789
Filling forms.....	802
Converting HTML contents to PDF.....	807
Merge PDF Documents	811
Open and Save PDF file in C# and VB.NET	813
Opening an existing PDF document.....	813
Opening an existing PDF document from Stream.....	814
Opening an existing PDF document from byte array.....	814
Opening an Encrypted PDF document.....	815
Opening a corrupted PDF document	816
Saving a PDF document to file system	818
Saving a PDF document to stream	820
Saving a PDF document into the same file or stream.....	821
Closing a document.....	823
Working with Document.....	824
Adding the document settings.....	824

Creating sections in a PDF	834
Printing PDF document	837
Working with document properties	838
Performing incremental update for PDF document	843
Choosing the viewer preferences	845
Adding document action.....	850
Working in Multi-Threading Environment.....	850
Uniform Resource Naming in PDF document	852
Memory Optimization.....	856
Find corrupted PDF document.....	858
Working with Pages	861
Adding a new page to the document.....	861
Inserting pages in a document.....	863
Adding margin to the PDF pages.....	865
Adding sections with different page settings	867
Get number of pages from a PDF document	874
Importing pages from an existing document.....	875
Importing pages from an existing document without bookmarks.	877
Rearranging pages in an existing document	880
Changing the page numbers in a PDF document.....	881
Removing pages from a document	884
Rotating a PDF page	886
Rotating an existing PDF page.....	888
Splitting a PDF file to individual pages.....	890
Working with Text.....	893
Drawing text in a new document.....	893
Drawing text in an existing document	895
Drawing text using different fonts	897
Measuring a string	905
Embedding fonts and working with Unicode text	908
Drawing Right-To-Left text.....	909
Adding a HTML Styled Text	913
Creating a multicolumn PDF document.....	915
Inserting Rich Text Format contents.....	922
Adding an Ordered List	923

Adding an Unordered List	927
Replace Fonts in an existing document	931
Search and get the bounds of a text in a document.....	932
Drawing complex script language text.....	932
Drawing text using OpenType font	938
Working with Images	941
Inserting an image in a new document.....	941
Inserting an image in an existing document	943
Inserting a vector image	949
Working with image masking.....	950
Replacing Images in an existing PDF document.....	952
Image Pagination	952
Applying transparency and rotation to the image.....	955
Converting multi page TIFF to PDF.....	958
Working with Brushes	960
Solid Brush	960
Linear gradient brush.....	963
Radial Gradient Brush	965
Tiling Brush.....	967
Working with Tables	969
Creating a simple table	969
Support for cell customization	991
Support for rows and columns customization	1005
Support for table customization	1021
Built-in table styles.....	1029
Pagination	1041
Adjust table width automatically	1048
Adding multiple tables	1052
String formatting.....	1057
Row and Column spanning	1079
Table cell styles	1086
Table row style	1092
Difference between PdfLightTable and PdfGrid	1098
Working with flow layout.....	1098
Working with Text.....	1099

Working with text and images	1103
Working with table.....	1109
Flow model using PdfLayoutResult	1115
Working with Forms.....	1122
Creating a new PDF form	1122
Set appearance to the PDF form fields	1164
Modifying the existing form field in PDF document	1166
Retrieving/Modifying the fore and back color of an existing form fields	1169
Filling form fields in an existing PDF Document	1172
Fill the XFA form fields along with Acroform in a same API	1194
Removing editing capability of form fields	1196
Removing the form fields from existing PDF document.....	1206
Importing FDF file to PDF	1208
Export PDF file to FDF	1209
Complex script support for form fields	1209
Auto naming of form fields	1218
Adding actions to form fields.....	1222
Auto resizing text box field text	1222
Troubleshooting.....	1224
Working with XFA	1229
Creating a new XFA form	1229
Creating dynamic XFA forms.....	1240
Creating static XFA forms.....	1243
Creating XFA form fields	1246
Working with XFA form flow directions.....	1316
Rotating XFA form fields	1322
Validating the date time field	1325
Customizing the numeric field value.....	1328
Adding nested sub forms	1331
Formatting XFA form fields	1335
Filling form fields in an existing XFA document.....	1349
Flattening XFA Form fields	1376
Removing the dynamic form fields from an existing XFA document.....	1376
Modifying the existing fields in XFA document	1379
Clear XFA date time field value	1381

Supported fields for XFA form	1383
Merge Documents	1384
Merging multiple documents from disk and stream	1384
Importing pages from multiple documents	1389
Best practices	1395
Optimizing PDF resources when merging PDF documents.....	1398
Working with Text Extraction	1401
Working with basic text extraction	1401
Working with layout based text extraction	1404
Text Extraction with Bounds	1405
Working with Image Extraction	1409
Image informations	1410
Converting HTML to PDF.....	1411
Steps to download the HTML converter installer	1411
Getting Started.....	1412
Supported and Unsupported Features by Rendering Engines.....	1415
Working with Document Conversions	1417
Converting Word documents to PDF	1417
Converting Excel documents to PDF	1422
Converting RTF documents to PDF	1427
Converting TIFF to PDF.....	1431
Converting XPS document to PDF.....	1435
Converting PDF to Image	1438
MHTML to PDF	1439
HTML to MHTML.....	1440
HTML to Raster Image.....	1441
HTML string to Raster Image	1442
Partial webpage to Raster Image	1444
HTML to SVG	1445
Partial webpage to SVG.....	1446
Working with Optical Character Recognition (OCR)	1448
Prerequisites and setting up the Tesseract Engine.....	1448
Performing OCR for an entire document.....	1448
Performing OCR with tesseract version 3.05	1449
Performing OCR for a region of the document.....	1450

Performing OCR on image.....	1451
Performing OCR for large PDF documents.....	1452
Performing OCR on rotated page of PDF document.....	1452
Layout result from OCR.....	1453
Native call.....	1454
Customizing temp folder.....	1456
Best Practices	1457
Troubleshooting	1458
Working with Hyperlinks.....	1458
Working with Web navigation	1458
Working with internal document navigation.....	1464
Working with external document navigation	1470
Working with PDF Templates.....	1473
Creating a new PDF template	1473
Creating templates from existing PDF document.....	1478
Working with PdfPageTemplateElement.....	1481
Creating Document Overlays	1485
Working with Headers and Footers	1489
Adding an automatic field in header and footer.....	1489
Working with Shapes	1493
Adding Shapes to a PDF document.....	1493
Working with shape pagination	1535
Working with Bookmarks.....	1537
Adding Bookmarks in a PDF	1537
Adding Bookmarks in an existing PDF document	1539
Adding a Child to the Bookmarks.....	1542
Inserting Bookmarks in an existing PDF.....	1545
Removing Bookmarks from an existing PDF	1547
Modifying the Bookmarks.....	1549
Working with Named Destination	1551
Adding Named Destination to a PDF document.....	1552
Adding Named Destination to an existing PDF document.....	1554
Removing/Modifying the named destination.....	1557
Adding named destination to the bookmarks	1559
Working with Annotations.....	1562

Adding annotations to a PDF document	1562
Flatten annotation	1567
Supported annotation types	1575
Measurement Annotations	1608
Modifying the annotations	1618
Removing annotations from an existing PDF	1621
Importing annotations from FDF file	1623
Importing annotations from XFDF file	1625
Exporting annotations to FDF file	1627
Exporting annotations to XFDF file	1629
Adding comments and review status to the PDF annotation	1630
Removing comments and review status from PDF annotation	1644
Modifying comments and review status	1649
Retrieve review status and comments from PDF annotation	1655
Printing Annotations	1659
Add Custom Stamp using Rubber Stamp Annotation	1662
Working with Attachments	1668
Adding attachment to a PDF document	1668
Removing attachment from an existing PDF	1673
Extracting and saving an attachment to the disk	1675
Working with Security	1676
Working with RC4 Encryption	1677
Working with AES Encryption	1682
Encryption Options	1688
Opening an encrypt-only-attachment document	1697
Set user password using event when accessing the attachment	1699
Protect attachments in existing PDF document	1701
Protect an existing document	1703
Changing the password of the PDF document	1705
Change the permission of the PDF document	1707
Remove password from the user password PDF document	1709
How to determine whether the PDF document is password protected or not?	1711
How to determine whether the PDF document is protected by user or owner password	1712
Working with PDF Redaction	1713
Removing sensitive content from the PDF document	1713

Display text on the redacted area.....	1714
Drawing image on the redacted area	1715
Drawing pattern on the redacted area	1716
Fill color on the redacted area	1718
Redaction without fill color and appearance.....	1719
Working with Digital Signature	1720
Adding a digital signature	1720
Adding a digital signature using stream.....	1724
Adding a digital signature using X509Certificate2	1730
Signing an existing document	1733
Sign an existing document using stream	1735
Externally sing a PDF document.....	1738
Create Long Term Validation (LTV) when signing PDF documents externally.....	1741
Adding a signature validation appearance based on the signature	1744
Adding a timestamp in digital signature	1748
Adding a timestamp to PDF document.....	1751
Adding a timestamp to existing PDF document.....	1753
Retrieve certificate details from an existing signed PDF document.....	1755
Enable Long Term Validation (LTV) PDF signature.....	1758
Adding a digital signature with customization.....	1762
Digital signature validation	1769
Working with Barcode	1776
Adding a one dimensional barcode to the PDF document	1776
Adding a two dimensional barcode to a PDF document	1778
Set location and size to the barcode.....	1780
Export Barcode as Image	1783
Customizing the barcode appearance	1785
Supported barcode types.....	1785
Working with Actions.....	1786
Adding an action to the PDF	1786
Supported action types.....	1787
Adding an action to the form field.....	1803
Adding an action to the bookmarks.....	1806
Working with Watermarks.....	1808
Adding text watermark in PDF document.....	1808

Adding image watermark in PDF document	1814
Working with Portfolio.....	1820
Creating a PDF portfolio.....	1820
Extracting file from PDF Portfolio	1823
Removing files from PDF Portfolio.....	1824
Working with Layers	1826
Adding Layers in a PDF document	1826
Adding annotation to layer	1832
Nested Layers.....	1837
Removing layers from an existing PDF document	1840
Flattening the layers in an existing PDF document.....	1842
Toggling the visibility of layers.....	1844
Tagged PDF.....	1849
Introduction	1849
Adding tag to text element	1849
Adding tag to image	1853
Adding tag to shapes.....	1856
Adding tag to Form Fields	1859
Adding tag to Annotation.....	1862
Adding tag to Hyperlink	1865
Adding tag to Template	1869
Adding tag to Table	1872
Adding tag to List Element	1877
Marking PDF content as an artifact	1882
Tag Reading Order	1887
Auto Tagging a new document	1892
How to pass accessibility full check	1895
Tagged PDF support for converting HTML to PDF	1896
Converting Word document to Tagged PDF	1896
Validating tagged PDF in Acrobat	1898
Working with Compression.....	1899
Compressing existing PDF document.....	1899
Compressing images with image quality	1900
Optimizing embedded font.....	1900
Optimizing page contents	1901

Remove metadata information.....	1902
Compressing the PDF content.....	1902
Compressing images	1904
Working with PDF Conformance.....	1906
PDF/A-1b conformance.....	1906
PDF/A-2b conformance.....	1908
PDF/A-3b conformance.....	1911
PDF/A-1a conformance.....	1914
PDF/A-2a conformance.....	1917
PDF/A-3a conformance.....	1919
PDF/A-2u conformance.....	1923
PDF/A-3u conformance.....	1925
PDF/X-1a conformance	1929
PDF to PDF/A-1b conversion.....	1930
Working with ZUGFeRD invoice	1930
Generating ZUGFeRD invoice.....	1930
Adding ZUGFeRD structured data as attachment.....	1932
Complete code	1933
Extract ZUGFeRD invoice from PDF	1936
Validating ZUGFeRD invoices using Adobe Acrobat	1938
Working with Metadata (XMP)	1942
Working with the XMP metadata	1942
Adding XMP metadata in a PDF document.....	1942
Adding XMP metadata in an existing PDF document	1945
Supported Schema types	1947
Basic Schema.....	1947
Adding Custom Schema to the PDF document.....	1963
Adding Custom Metadata to the PDF document.....	1965
Removing Custom Metadata from an existing PDF document.....	1966
Working with image metadata	1968
Adding XMP metadata along with an image in a PDF document	1968
Extracting XMP metadata from PDF image	1970
Working with Color Spaces	1971
Working with color space in document	1971
Device Color Space.....	1971

CIE-based Color Spaces	1971
ICC-based Color Spaces	1977
Pantone colors	1985
Working with color space in graphics	1991
Working with JavaScript.....	1998
Document level JavaScript action	1998
JavaScript action to the form fields	2000
JavaScript in 3D Annotation	2003
Supported and Unsupported Features	2005
Presentation.....	2012
Overview of PowerPoint Presentation	2012
Key features	2012
Compatible Microsoft PowerPoint Versions.....	2013
Assemblies Required.....	2013
Converting PowerPoint Presentation to PDF.....	2013
NuGet Packages Required.....	2014
Converting PowerPoint Presentation into PDF.....	2015
Converting PowerPoint Presentation to image	2015
Converting charts in Presentation	2016
NuGet package installation and uninstallation.....	2016
Getting Started.....	2019
Creating a simple PowerPoint Presentation with basic elements from scratch.....	2019
Converting PowerPoint Presentation to PDF.....	2027
Document Object Model representation	2030
Load and save the Presentation.....	2031
Opening an existing Presentation from file system	2031
Opening an existing Presentation from stream	2031
Opening an encrypted Presentation	2032
Saving a PowerPoint Presentation to file system	2033
Saving a PowerPoint Presentation to stream	2035
Sending to a client browser	2036
Closing a PowerPoint Presentation.....	2037
Working with PowerPoint presentation	2038
Cloning a PowerPoint presentation	2038
Printing a PowerPoint presentation	2041

Working with PowerPoint presentation properties	2044
Custom Document properties.....	2046
Marking a PowerPoint presentation as final	2050
Working with Slides in PowerPoint.....	2052
Adding slide to the PowerPoint presentation.....	2052
Create a slide with predefined LayoutSlide	2053
Adding Custom layout slide	2059
Cloning slide	2064
Merging slide.....	2067
Removing slide	2069
Converting to image.....	2071
Changing Slide background	2073
Create and edit Master and Layout slides	2075
Access the MasterSlide	2076
Create a custom LayoutSlide.....	2076
Working with Paragraph	2077
Adding Paragraph to slide.....	2077
Applying Paragraph formatting.....	2080
Working with text	2082
Modifying text.....	2087
Enabling shrink text on overflow option.....	2089
Removing the paragraph	2090
Working with lists	2092
Creating a simple list.....	2092
Creating a Multilevel List	2109
Working with Shapes	2116
Adding shapes to a slide	2116
Iterating through shapes.....	2118
Specifying shape properties.....	2120
Removing the shapes.....	2124
Working with GroupShape.....	2126
Working with images	2133
Adding Images.....	2133
Replacing Images	2135
Removing Images	2138

Working with PowerPoint Tables	2140
Create a table by adding rows	2140
Create a table by adding columns.....	2143
Append a new row at the end of table	2146
Copy an existing row to the end of table	2148
Insert a row in table	2150
Append a new column at the end table.....	2152
Copy an existing column to the end of table	2154
Insert a column to a table	2156
Get the actual height of the table.....	2158
Applying table formatting	2159
Applying table styles	2164
Modifying the table.....	2168
Merging the cells.....	2171
Removing the table	2174
Working with Charts	2176
Creating a Chart from scratch	2176
Creating charts from excel sheet	2181
Creating Custom Charts	2183
Refreshing the chart.....	2189
Editing the Chart Data.....	2191
Customizing the chart	2194
Applying 3D Formats.....	2211
Chart to Image conversion.....	2214
Removing the chart from slide.....	2215
Creating a Scatter chart	2217
PowerPoint 2016 Charts	2220
Supported Chart Types	2250
Working with Animations	2252
Adding animation effect to shapes	2252
Adding interactive animation	2254
Adding animation to text	2257
Adding exit animation effect.....	2259
Edit existing animation effect	2262
Modifying animation effect sub type.....	2264

Modifying timing of animation effect	2266
Reordering the animation effects	2269
Creating custom path animation effect	2271
Removing animation effect	2275
Supported animation effects type	2278
Add and edit transitions in PowerPoint slides	2285
Set a transition effect to a PowerPoint slide	2285
Modify a transition effect applied to a PowerPoint slide	2287
Set the transition duration	2289
Set the transition delay	2292
Set the trigger mode for the transition	2294
Set the speed for transition effect	2296
Supported transition effect types:	2298
Adding connectors in PowerPoint slides	2303
Adding a single point connector	2306
Editing a connector in PowerPoint slide	2308
Updating the positions of the connector in PowerPoint slide	2312
Removing a connector from PowerPoint shapes	2314
Working with Headers and Footers	2316
Add Headers and Footers in PowerPoint	2316
Modify Headers and Footers in PowerPoint	2335
Remove Headers and Footers from Title Slides	2343
Working with Notes	2346
Adding Notes to a Slide	2346
Adding Text into the Notes	2348
Adding a numbered list to Notes	2351
Removing Notes from a Slide	2357
Working with SmartArt	2358
Adding SmartArt to a Slide	2358
Adding a node to the SmartArt	2360
Modifying SmartArt appearance	2365
Iterating through child nodes of an existing SmartArt	2367
Removing node from an existing SmartArt	2370
Assistant nodes in SmartArt	2372
Limitations	2375

Supported SmartArt layout types	2375
Working with Comments	2378
Adding a comment	2378
Replying to a comment	2379
Modifying the comment	2382
Deleting the comment	2386
Working with Sections	2389
Creating a section	2390
Moving the sections within a PowerPoint presentation	2395
Moving a slide within sections	2397
Cloning and merging the slides in a section.....	2399
Removing a section	2401
Remove all sections	2403
Working with Macros.....	2404
Loading and Saving Macro enabled Presentation	2404
Removing Macros from Macro enabled Presentation.....	2406
Security	2408
Encrypting with password.....	2408
Decrypting the PowerPoint Presentation	2409
Write Protection	2410
Working with OLE Objects	2414
Inserting OLE Object to a Slide.....	2415
Presentation to image conversion	2417
.NET Framework.....	2417
Assemblies Required.....	2417
UWP	2421
Font substitution for unavailable fonts.....	2423
Presentation to PDF conversion	2425
Font substitution for unavailable fonts.....	2428
Show Warning for unsupported elements	2434
Handouts.....	2437
Notes pages.....	2438
Include hidden slides.....	2439
PDF Conformance	2439
Chart quality.....	2440

Optimizing the converted PDF document size.....	2441
PowerPoint to PDF conversion in Azure platform	2443
FAQ's	2446
XlsIO	2448
Overview	2448
Assemblies Required.....	2448
Converting Excel document to PDF.....	2449
Converting Excel Worksheet to Image.....	2450
Converting Excel Chart to Image.....	2450
NuGet Packages Required.....	2451
Converting Excel document into PDF.....	2452
Converting Excel Worksheet to Image.....	2453
Converting Charts in XlsIO	2454
NuGet Package Installation and Uninstallation	2456
Getting Started - Create Excel File in C# and VB.NET	2457
Create a Hello World Excel File	2458
Create a Simple Excel File	2461
Import Data to Excel Worksheets	2473
Export Data from Excel Worksheets	2480
Template based data filling using Template Markers.....	2485
Document Object Model.....	2493
Loading and Saving Workbook.....	2496
Opening an existing workbook	2496
Opening an existing workbook from Stream	2498
Saving a Excel workbook to file system	2499
Saving a Excel workbook to stream	2501
Sending to a client browser	2503
Closing a workbook.....	2504
Working with Excel Worksheet.....	2507
Create a Worksheet	2507
Access a Worksheet	2509
Remove a Worksheet.....	2512
Move or Copy a Worksheet	2513
Highlight Worksheet Tabs.....	2518
Freeze Panes	2519

Unfreeze Panes	2523
Split Panes	2525
Page Setup Settings.....	2528
Show or Hide Worksheet	2530
Activate a Worksheet.....	2532
Show or Hide Worksheet Tabs.....	2534
View Settings.....	2536
Open a CSV File	2541
Save Worksheet as CSV.....	2543
Save Worksheet as HTML	2547
Worksheet Rows and Columns Manipulation	2551
Insert Rows and Columns.....	2551
Delete Rows and Columns	2554
Show or Hide Rows and Columns	2558
Show or Hide Specific Range.....	2561
Adjust Row Height and Column Width	2563
Auto-Fit Rows and Columns.....	2568
Group or Ungroup Rows and Columns	2573
Worksheet Cells Manipulation.....	2581
Accessing a Cell or a Range	2581
Accessing Relative Range	2584
Accessing used range of a Worksheet	2592
Get Precedent and Dependent Cells or Range	2595
Clearing a Cell Content.....	2603
Copy or Move a Range	2605
Copy and Paste As Link	2610
Skip Blanks While Copying	2612
Find and Replace	2615
Data Sorting	2621
Data Filtering.....	2631
Hyperlinks	2658
Working with Cell or Range Formatting.....	2674
Create a Style	2675
Set Default Style for row or column.....	2677
Apply Global Style	2679

Apply Number Formats.....	2686
Hide Cell Content by setting Number Format.....	2699
Apply Cell Text Alignment.....	2701
Merging and Un-Merging Cells	2710
Apply Wrap Text.....	2713
Auto-Fit Rows or Columns	2715
Apply Font Settings	2718
Apply Color Settings.....	2722
Apply Border Settings	2724
HTML String Formatting.....	2729
Rich-Text Formatting.....	2731
Working with Data	2734
Importing Data to Worksheets	2734
Exporting from Worksheet to Data Table.....	2792
Exporting from Worksheet to Collection Objects.....	2794
Export data from Excel to Nested Class Objects.....	2798
Importing Data from Microsoft Grid Controls to Worksheet	2804
Working with Formulas.....	2807
Enable and Disable Calculation	2807
Writing a Formula	2808
Reading a Formula	2812
Accessing a Calculated value.....	2813
Applying Argument Separators Based on Cultures.....	2819
Array of Formula	2821
Incremental Formula.....	2823
External Formula.....	2825
Calculated Column	2827
Supported Functions.....	2830
Add-in Functions	2844
Defined Names.....	2847
Formula Auditing.....	2851
Calculation Engine.....	2854
Calculate Options.....	2854
Working with Conditional Formatting	2859
Create a Conditional Format.....	2859

Reading Conditional Formats in XlsIO	2867
Removing Conditional Formats.....	2870
Using FormulaR1C1 property in Conditional Formats	2876
Format Unique and Duplicate Values	2878
Format Top or Bottom Values.....	2886
Format Above or Below Average Values	2895
Advanced Conditional Format Types	2904
Working with Data Validation.....	2919
Text Length Validation	2919
Time Validation	2920
List Validation.....	2921
Number Validation	2922
Date Validation	2923
Custom Validation.....	2924
Working with Charts	2934
Creating a Chart	2934
Creating Custom Charts	2945
Remove a chart	2961
Chart Appearance Settings	2963
Fill Chart Elements with Picture.....	2992
Applying 3D Formats.....	2998
Customizing chart and Chart Elements.....	3003
Explode a Pie Chart	3012
Add Picture to Chart and assign Hyperlink	3015
Add DataTable to Chart.....	3018
Sparkline.....	3022
Excel 2016 Charts	3030
Supported Chart Types	3060
Working with Template Markers	3062
Template marker Syntax	3062
Bind from Array.....	3063
Bind from DataTable	3070
Bind from Collection Objects with images.....	3074
Bind from Nested Collection Objects with import data and group options	3080
Template marker with conditional formatting	3097

Template marker with Hyperlink	3109
Working with Tables	3117
Creating a table.....	3117
Accessing a table.....	3119
Formatting a table.....	3122
Insert or remove columns in a table	3133
Adding a total row.....	3135
Create a table from external connection.....	3138
Adding parameters to query in Excel table.....	3141
Working with Pictures.....	3149
Positioning and Re-Sizing Pictures	3151
Adding Images from External Link	3154
Adding SVG Images	3156
Working with Pivot Tables	3158
Create a pivot table.....	3158
Editing and formatting a pivot table	3164
Refresh a pivot table	3168
Expand or collapse rows in pivot table	3171
Applying pivot table filters.....	3174
Applying pivot table settings.....	3182
Sort by value in Pivot Table.....	3188
Other pivot table operations	3192
Working with Pivot Charts	3199
PivotChart Options.....	3201
Security	3202
Protect Workbook.....	3202
Protect Worksheet.....	3212
Protect Cell.....	3220
Working with Drawing Objects	3223
Form Controls	3223
Comments.....	3236
AutoShapes	3246
Group Shapes.....	3249
OLE Objects	3255
Working with Macros.....	3264

Creating a Macro.....	3264
Editing a Macro	3288
Removing Macros	3291
Excel to PDF Conversion.....	3302
Workbook to PDF	3302
Worksheet to PDF	3305
Excel with chart to PDF	3310
Substitute Font in Excel-to-PDF Conversion	3313
Excel to PDF conversion in Linux OS	3320
Supported elements.....	3325
Unsupported elements	3325
Excel to PDF Converter Settings.....	3326
Auto-detect Complex Script.....	3326
Custom Paper Size.....	3329
Display Gridlines.....	3332
Embed Fonts	3341
Export Bookmarks	3344
Export Document Properties	3347
Export Quality Image	3350
Header Footer Option	3353
Convert Blank Page	3358
Convert Blank Sheet.....	3361
Layout Options	3364
PDF Conformance Level	3382
Template Document	3385
Throw When Excel File Is Empty	3389
Capture Warnings in Excel-to-PDF Conversion.....	3391
Worksheet to Image conversion.....	3397
Assemblies Required.....	3397
Convert as bitmap.....	3398
Save as stream	3398
Chart to image conversion.....	3402
Supported chart types	3405
Supported chart elements	3406
Excel to ODS Conversion	3407

Saving ODS in different platforms.....	3407
Supported and unsupported elements in ODS conversion	3411
Custom XML Support	3412
Migration from Office Automation to Syncfusion XlsIO	3417
Why Syncfusion's XlsIO over Office Automation?	3417
Topic links.....	3417
Supported Features by File Formats.....	3418
Supported Features by Platforms	3421
Supported Features by .NET Framework and .NET Standard	3421
Supported Features in Blazor Platform.....	3426
Improving Performance	3428
UsedRange	3428
Range Access.....	3428
Styles	3430
AutoFit.....	3431
Importing DataTable	3431
Data Validation.....	3432
Known Exceptions Details.....	3432
ApplicationException	3432
ArgumentException	3433
ArgumentNullException.....	3435
ArgumentOutOfRangeException	3435
ExcelWorkbookNotSavedException.....	3441
FileNotFoundException.....	3441
InvalidRangeException.....	3442
NotSupportedException.....	3442
FAQ Section.....	3443
How to open an existing XLSX workbook and save it as XLS?.....	3443
How to open an Excel file from Stream?	3443
How to save a file to stream?	3444
How to create and open Excel Template files by using XlsIO?	3445
How to open an Excel 2013 Macro Enabled Template?	3446
How to change the grid line color of the Excel sheet?	3446
How to copy and paste the values of the cells that contain only formulas?	3447
How to copy a range from one workbook to another?	3448

How to merge several excel files from more than one workbook to a single file?	3449
How to ignore the green error marker in worksheets?	3450
How to protect certain cells in a worksheet?	3450
How to set a line break inside a cell?	3451
How to set or format a Header/Footer?	3452
How to set print titles?	3452
How to unfreeze the rows and columns in XlsIO?	3454
What is the maximum range of Rows and Columns?	3454
How to use Named Ranges with XlsIO?	3454
How to add chart labels to scatter points?	3455
How to create a Chart with a discontinuous range?	3456
How to define discontinuous ranges?	3458
How to format text within a cell?	3459
How to hide the summary rows and columns using XlsIO?	3460
How to zip files using the Syncfusion.Compression.Zip namespace?	3461
How to zip all the files in subfolders using the Syncfusion.Compression.Zip namespace?	3462
How to protect the zip files with password using Syncfusion.Compression.Base?	3466
How to un-protect the zip files using Syncfusion.Compression.Base?	3466
Does Essential XlsIO provide support for Client Profile?	3467
How to resolve the "File does not contain workbook stream" error in Syncfusion.XlsIO.Base.dll?	3467
How to resolve "Excel cannot open the file 'filename.xlsx' because the file format for the file extension is not valid. Verify that the file has not been corrupted and that the file extension matches the format of the file" error?	3468
What happens when an Excel file containing uninstalled fonts is converted to PDF/Image?	3469
How to avoid exception when adding worksheets with same name	3470
How to change data point label color of a Waterfall chart	3472
How to check whether an Excel document contains macro?	3474
Does XlsIO support password protected macro in the Excel documents?	3477
Does XlsIO support Excel files with macros that are digitally signed?	3477

Welcome to Syncfusion Essential File Formats Platform

Essential File Formats is a collection of .NET class libraries to create, edit, write, and convert PDF, Excel, Word, and PowerPoint file formats in .NET Framework [C#, VB.NET], .NET Core, UWP, and Xamarin applications without Microsoft Office or Adobe dependencies.

How to best read this user guide

- The best way to get started would be to read the “Getting Started” section for the component you would like to start first. The “Getting Started” and “Assemblies required” sections gives enough information, so it is recommended to read these sections end-to-end before starting to write a code. All other information can be referred as needed.
- After learning the basics about the component, integrate the component into your application. A good starting point is to refer to the code examples in the sample browser and in this user guide. It is very likely that you can find a code example that resembles your intended usage scenario.
- Another valuable resource is available in the [API reference](#) that provides detailed information on the object hierarchy as well as the settings available on every object.

Additional help resources

The [Knowledge Base](#) section contains responses for common questions asked by the customers. This would be a good place to search for the topics that are not covered in the User Guide.

Similar to the Knowledge Base, the [Forum](#) section also contains responses to the questions that are asked by other customers.

Create a support incident

If you are unable to find the information you are looking for in the self-help resources mentioned above, please contact us by creating a [support ticket](#).

List of File Format libraries:

File Format library	Description
Essential PDF	.NET Class library used to create, read, and write PDF files in .NET Framework [Windows Forms, WPF, ASP.NET MVC, ASP.NET], .NET Core, UWP and Xamarin applications.
Essential XlsIO	.NET Class library used to create, read, edit, write, and convert Microsoft Excel files in .NET Framework [Windows Forms, WPF, ASP.NET MVC, ASP.NET], .NET Core, UWP, and Xamarin applications.
Essential DocIO	.NET Class library used to create, read, edit, and convert Microsoft Word files in .NET Framework [Windows Forms, WPF, ASP.NET MVC, ASP.NET], .NET Core, UWP, and Xamarin applications.
Essential Presentation	.NET Class library used to create, read, edit, and convert Microsoft PowerPoint (PPTX) files in .NET Framework [Windows Forms, WPF, ASP.NET MVC, ASP.NET], .NET Core, UWP, and Xamarin applications.

Quick Start links:

[Create a PDF file in C# without Adobe](#)

[Create Excel file in C# without Microsoft Office](#)

[Create Word file in C# without Microsoft Office](#)

[Create PowerPoint file in C# without Microsoft Office](#)

System Requirements

The following topic describes the system requirements needed by Syncfusion File Formats platform.

Operating Systems

Syncfusion File Formats platform can be installed on any of the following operating systems:

- Windows 10
- Windows 8, 8.1
- Windows 7
- Windows Vista
- Windows Server 2008 and later

Hardware Environment

- Processor: x86 or x64
- RAM : 512 MB (minimum), 1 GB (recommended)
- Hard disk: up to 8.5 GB of available space may be required. However, 750 MB free space is required in boot drive even if you are installing in other drive.

Development Environment

In order to develop applications with Syncfusion File Formats platform, you need to have installed anyone of the Visual Studio version.

- Microsoft Visual Studio 2005
- Microsoft Visual Studio 2008
- Microsoft Visual Studio 2010
- Microsoft Visual Studio 2012
- Microsoft Visual Studio 2013
- Microsoft Visual Studio 2015
- Microsoft Visual Studio 2017

Download Installer

You can download the installer from Syncfusion.com website.

Download the Trial Version

There are two ways to download our 30-day trial.

Free Trial Page

1. You can evaluate our 30-day free trial from [Free Trial](#) page.

2. Once you fill the required form or made the login using the your Syncfusion registered account you can download the trial installer in the confirmation page.
3. You can download the latest version trial installer.
4. You can unlock the installer using the unlock key, also you can unlock the installer using the Syncfusion registered login credential.
5. You can download the trial installer using the [Trials & Downloads](#) page under your registered account at any time before the trial expire. (Refer the below screenshot).

Trial Downloads and Unlock Keys

Start Trial Page

1. You can evaluate our 30-day free trial from [Start Trial](#) page.
2. You should login using your Syncfusion account to access this page.
3. You can start your trail by clicking on the required product.

Note: If you already using the trail products and it's not expired, you couldn't start the trial again for same product.

4. After you started the trial, you can download the latest version trial installer using the [Trials & Downloads](#) page.
5. In [Trials & Downloads](#) page, you can find your current active trial products. Trials, which you done in both Free Trial Page and Start trial pages are listed here.
6. Use the Download (element 1 in below screenshot) button to download the installer of respective product.
7. Online installer can be downloaded from this page.
8. No need of unlock key to unlock the online installer.
9. You can unlock the installer using the Syncfusion registered login credential.

License Downloads and Unlock Keys

Note: You can generate the license key for your active trial products from [Trials & Downloads](#) page. This license key will be mandatory to use our trial products in your application. To know more about License key, refer this [help topic](#).

Download the License Version

1. You can find your available licensed products which under your registered Syncfusion account in [License & Downloads](#) page.
2. You can find all the licenses (both active licenses and expired licenses) which are under your account.
3. You can find the licenses listed based on SKU names.
4. Use the Download (element 1 in below screenshot) button to download the installer of respective product.
5. Latest version installer will be downloaded from this page.
6. You can navigate to [Downloads Older Versions](#) (element 2 in below screenshot) to download older version installers.
7. From 16.2 version online installer will be downloaded by default, and earlier versions offline installer will be downloaded.

Note: Online Installer have been introduced from the release version 16.2.

8. You can navigate to More Downloads Options (element 3 in below screenshot) to download other installers.
9. EXE and Zip format available to download for Windows OS. Both are Offline Installer.
10. No need of unlock key to unlock the online installer.
11. You can unlock the installer using the unlock key for versions earlier to 16.2, also you can unlock the installer using the Syncfusion registered login credential.

License Downloads and Unlock Keys



Note: You can generate the license key for your licensed products from [License & Downloads](#) page. This license key will be required only from release version 16.2. To know more about License key, refer this [help topic](#).

Installation and Upgrade

Installation using Web Installer

Overview

Starting with version 16.2 (2018 Vol 2), Syncfusion provides Web Installer for Essential Studio platforms. This installer reduces the burden of downloading the installer of larger size. You can just download and launch the online installer which will be of smaller size and it will download and install the Essential Studio products you have selected. The Essential Studio Web Installer includes the following platforms. You can download the latest version Essential Studio Web Installer [here](#).

Starting with the version 17.3 (2019 Vol 3), Syncfusion provides updated version of Web Installer which allows both installation and uninstallation of the platforms for that specific version.

Web (Essential JS 2)

- ASP.NET MVC
- ASP.NET Core
- JavaScript

Mobile

- Xamarin
- Flutter

Desktop

- Windows Forms
- WPF
- Universal Windows Platform

FileFormats

- Read and Write Excel, Word, PDF and PowerPoint files

Web (Essential JS 1)

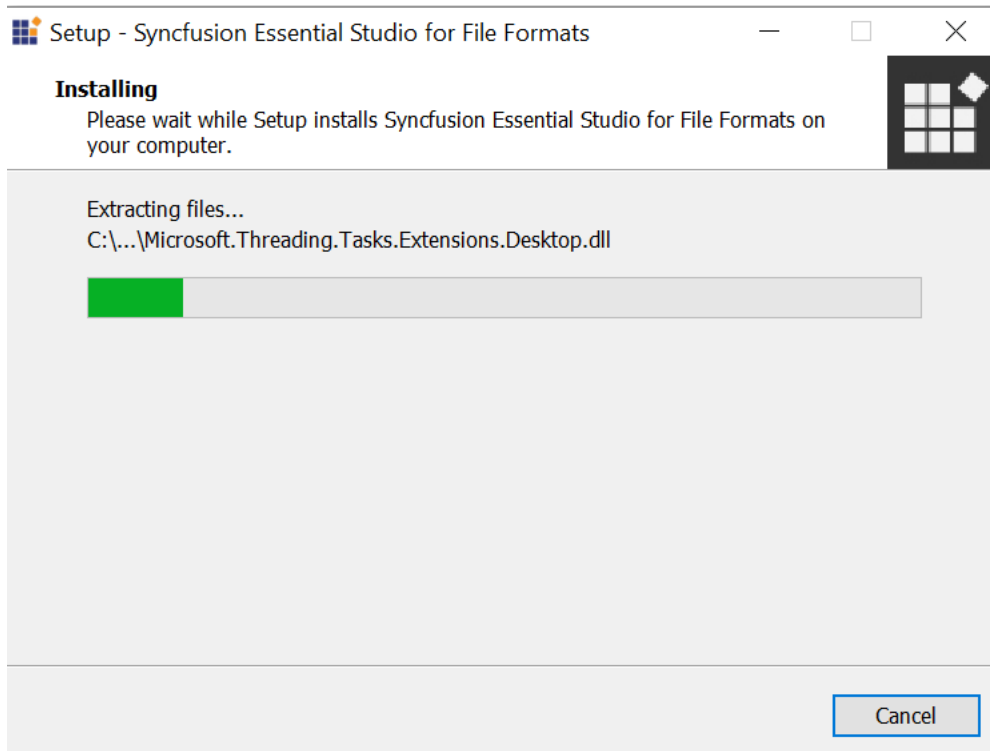
- ASP.NET Web Forms
- ASP.NET MVC
- ASP.NET Core
- JavaScript
- PHP
- JSP

Note: Universal Windows Platform will be installed in Windows 8.1 and later.

[Installation](#)

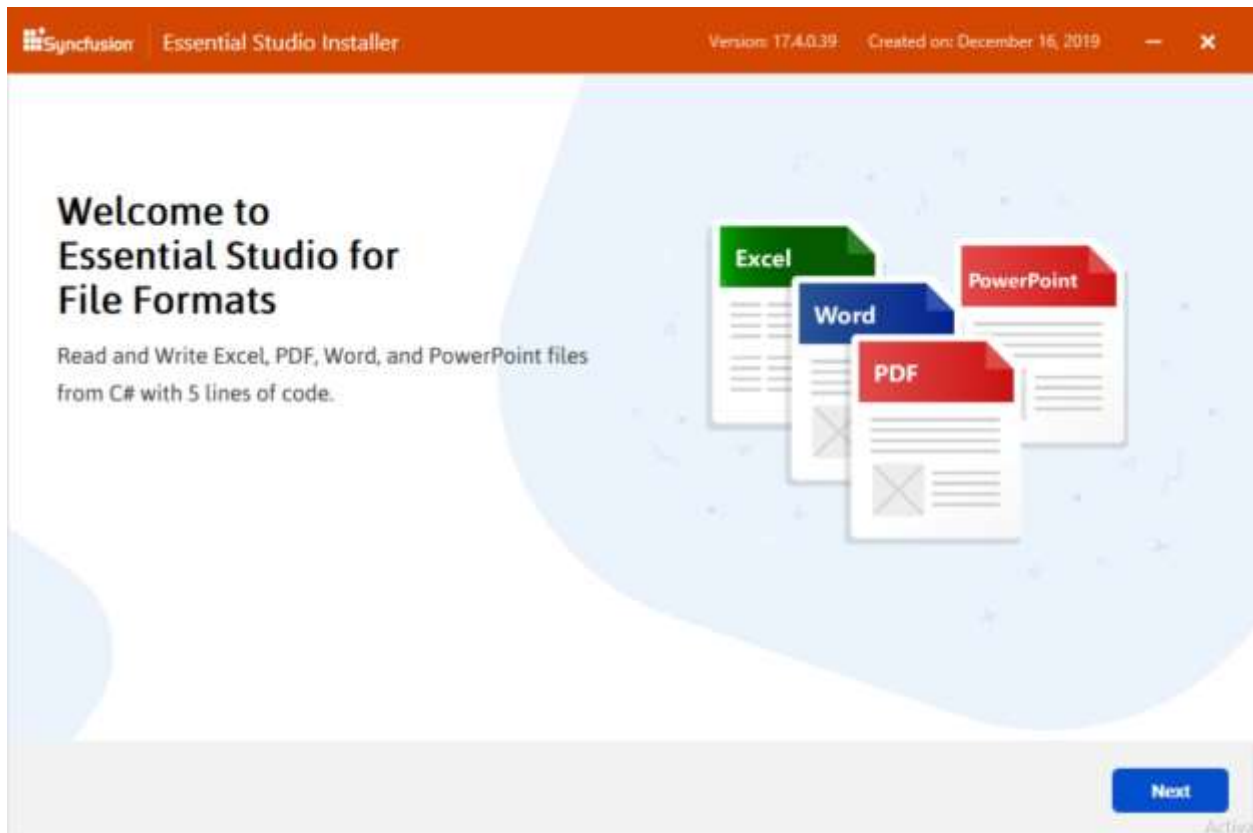
The following procedure illustrates how to install Essential Studio Platform Online Installer.

1. Double-click the Syncfusion Essential Studio Platform Online Installer file. The installer Wizard opens and extracts the package automatically.

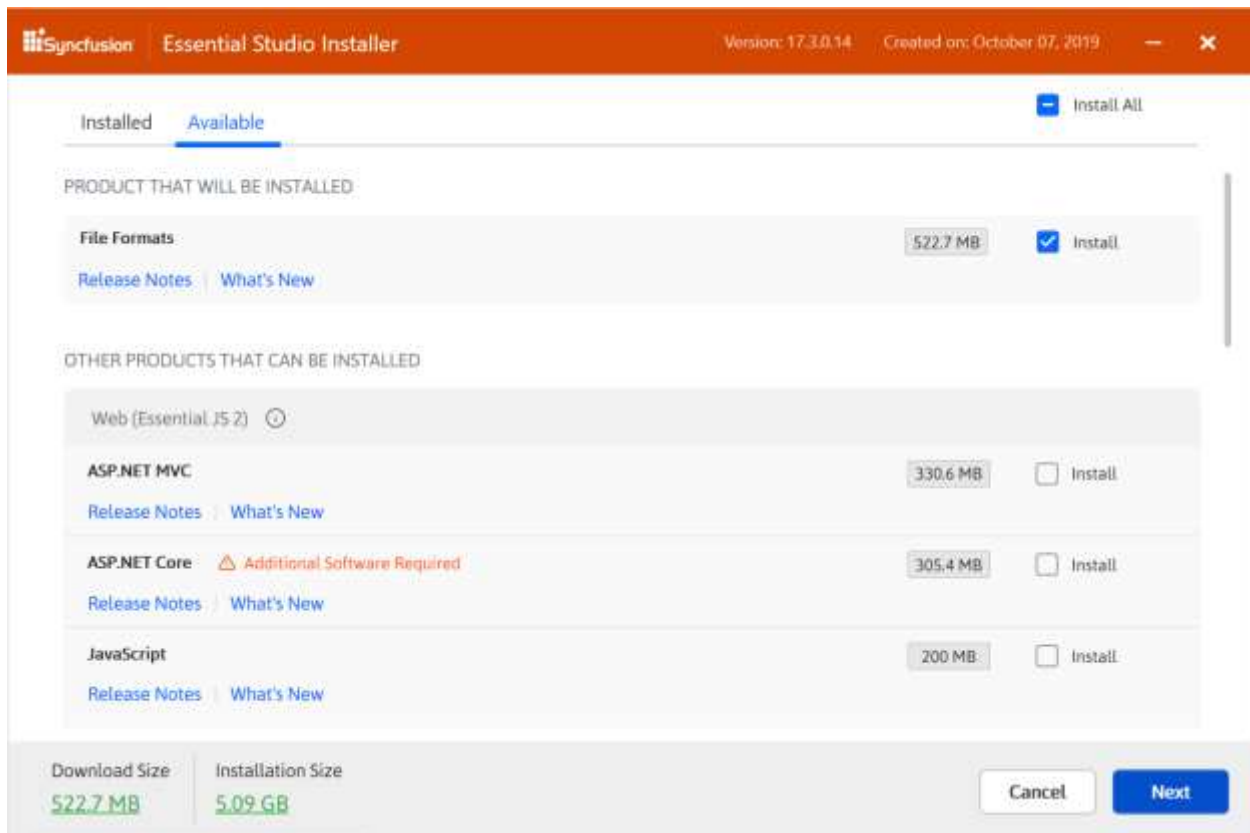


Note: The installer wizard extracts the `syncfusionfileformatswebinstaller_{version}.exe` dialog, displaying the unzip operation of the package.

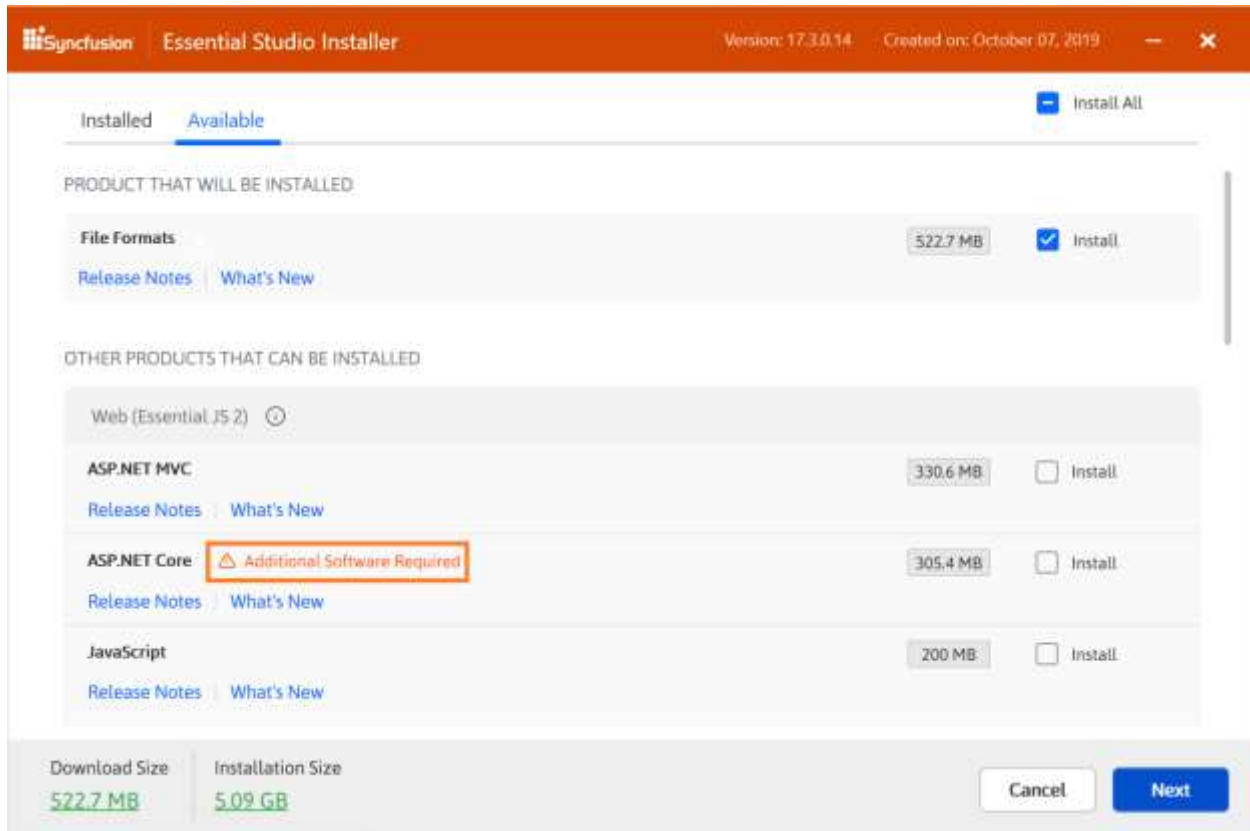
2. Welcome wizard of the Syncfusion Online Installer will be displayed. Click Next.



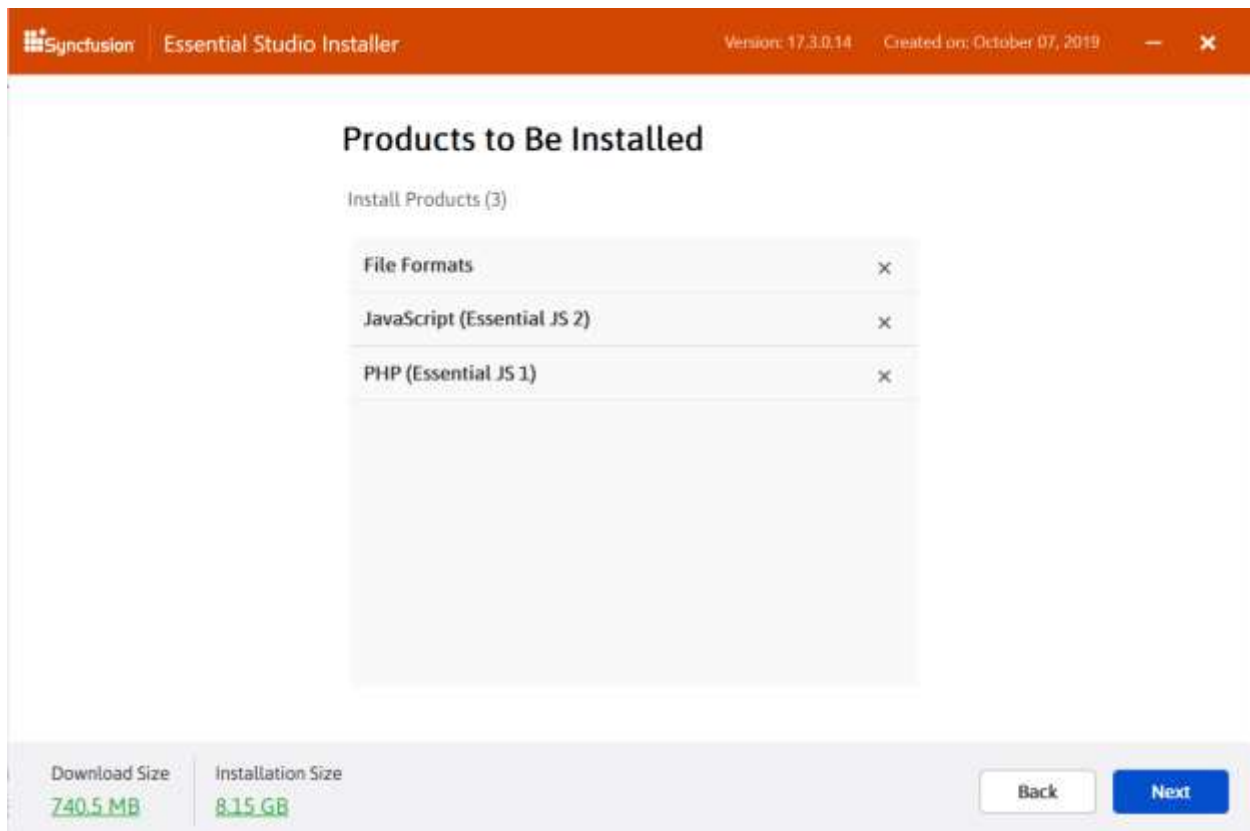
3. Platform Selection wizard will be displayed. Select the platforms to be installed from the **Available** tab. Select **Install All** checkbox to select all the platforms. Click Next.



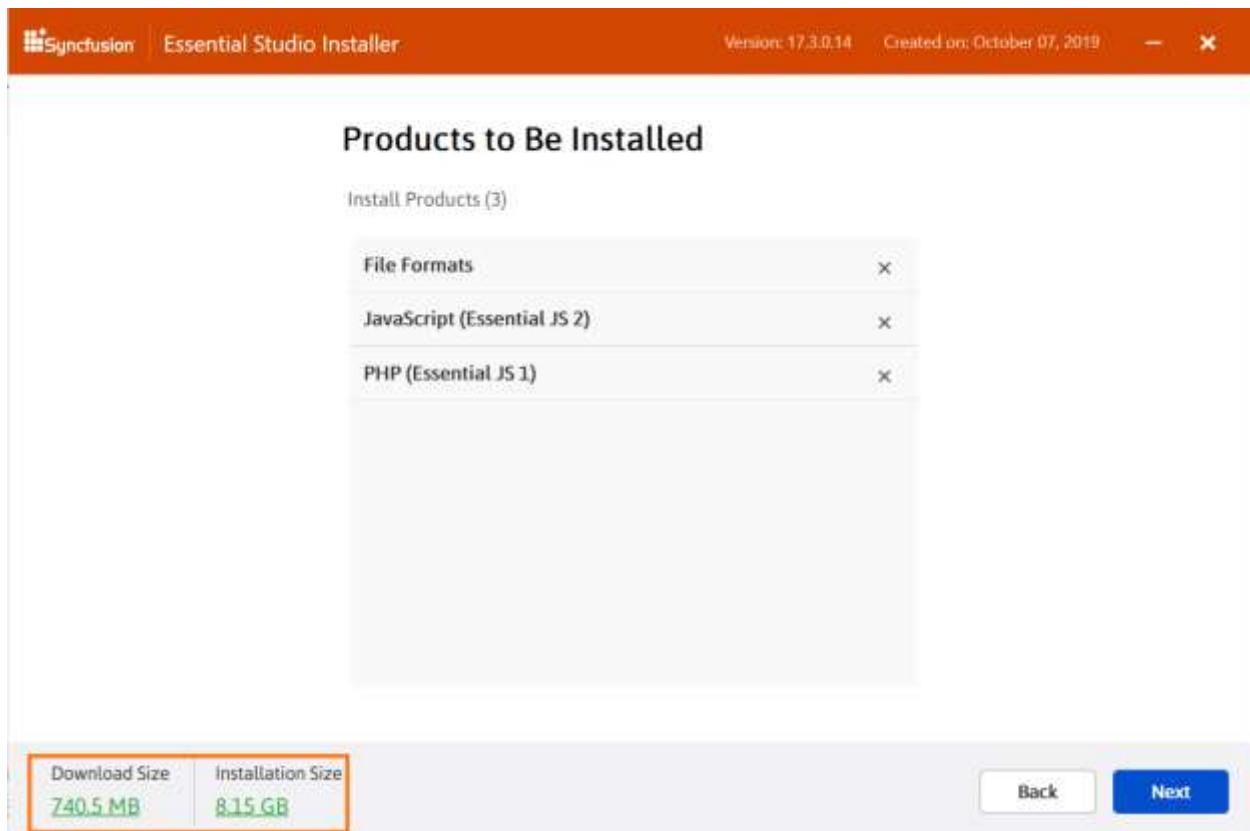
Information: If the required software of the selected platform was not already installed, **Additional Software Required** alert will be displayed. However, you can continue the installation and install the required software later.



4. Confirmation wizard will be displayed. Here you can view and modify the list of platforms that will be installed.



Note: You can check the Estimated size of the Download and Installation by clicking the **Download Size** and **Installation Size** link.



5. Configuration wizard will be displayed. Here you can change the Download, Install and samples location. Also, you can change the Additional settings by platform basis. To install using the default configuration, click Next.

Information: From version 17.3 (2019 Vol 3), Syncfusion provides option to provide custom download location.

Note: From the 2018 Volume 2 release, Syncfusion has changed the install and samples location

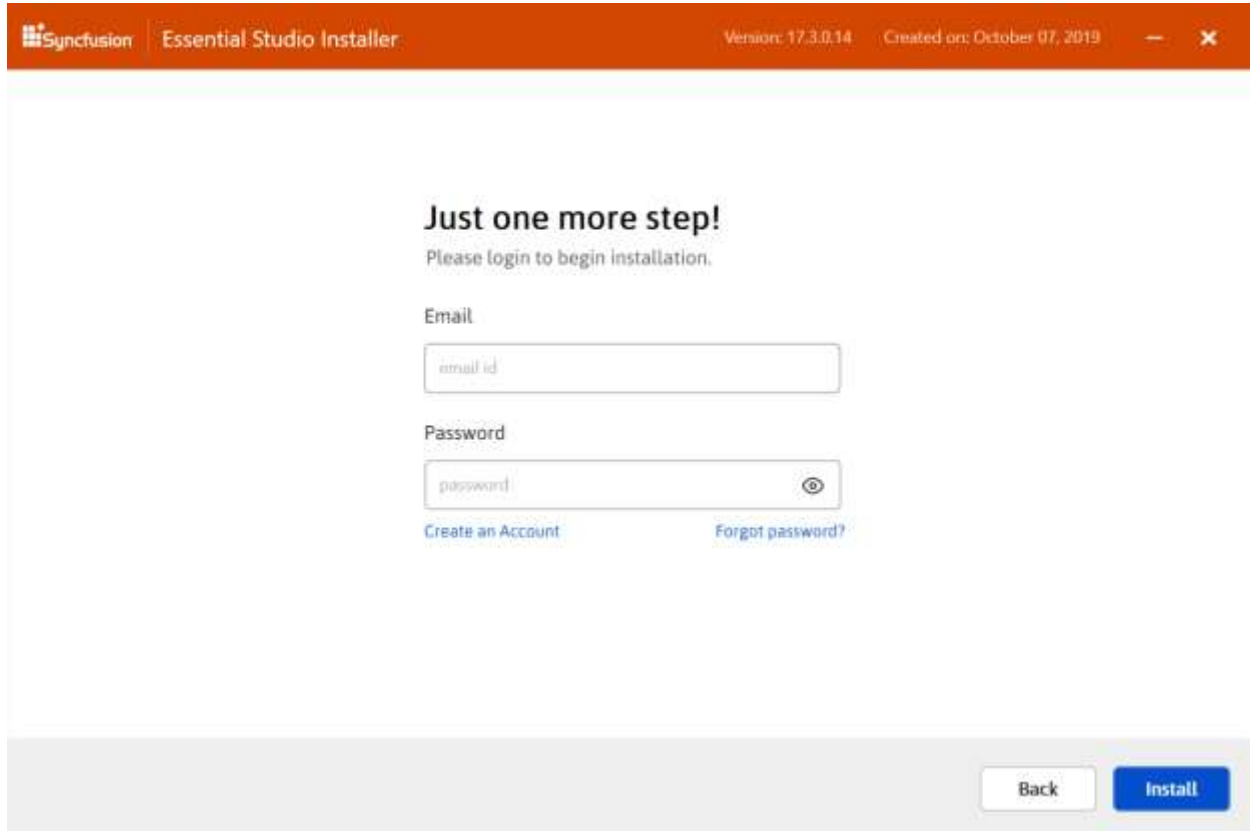
Default Install location: {ProgramFilesFolder}\Syncfusion\{Platform}\{version}

Default Samples location: C:\Users\Public\Documents\Syncfusion\{platform}\{version}

However, you can change the locations by clicking browse button.

- Select the **Install Demos** check box to install Syncfusion samples, or leave the check box clear, when you do not want to install Syncfusion samples.
- Select the **Register Syncfusion Assemblies in GAC** check box to install the latest Syncfusion assemblies in GAC, or clear this check box when you do not want to install the latest assemblies in GAC.
- Select the **Configure Syncfusion controls in Visual Studio** check box to configure the Syncfusion controls in the Visual Studio toolbox, or clear this check box when you do not want to configure the Syncfusion controls in the Visual Studio toolbox during installation. Note that you must also select the Register Syncfusion assemblies in GAC check box when you select this check box.
- Select the **Configure Syncfusion Extensions controls in Visual Studio** checkbox to configure the Syncfusion Extensions in Visual Studio or clear this check box when you do not want to configure the Syncfusion Extensions in Visual Studio.
- Select the **Create Desktop Shortcut** checkbox to create the desktop shortcut for Syncfusion Control Panel.

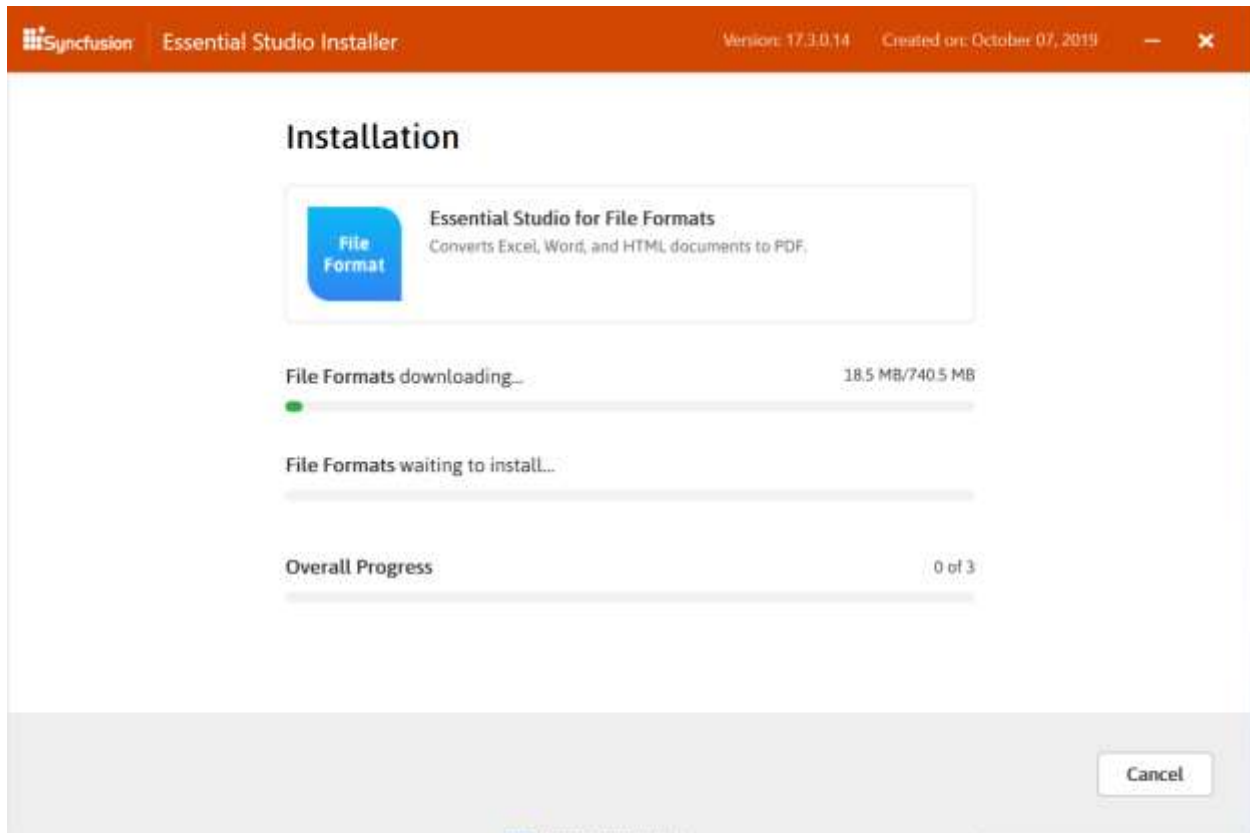
6. After reading the License Terms and Conditions, check the **I agree to the License Terms and Privacy Policy** check box. Click Next.
7. Login wizard will be displayed. You should enter your Syncfusion Direct-Trac login credentials. If you don't have Syncfusion Direct-Trac login credentials, then you can click on **Create an Account**. Else if you forgot your password, click on **Forgot Password** to create new password. Click Install.



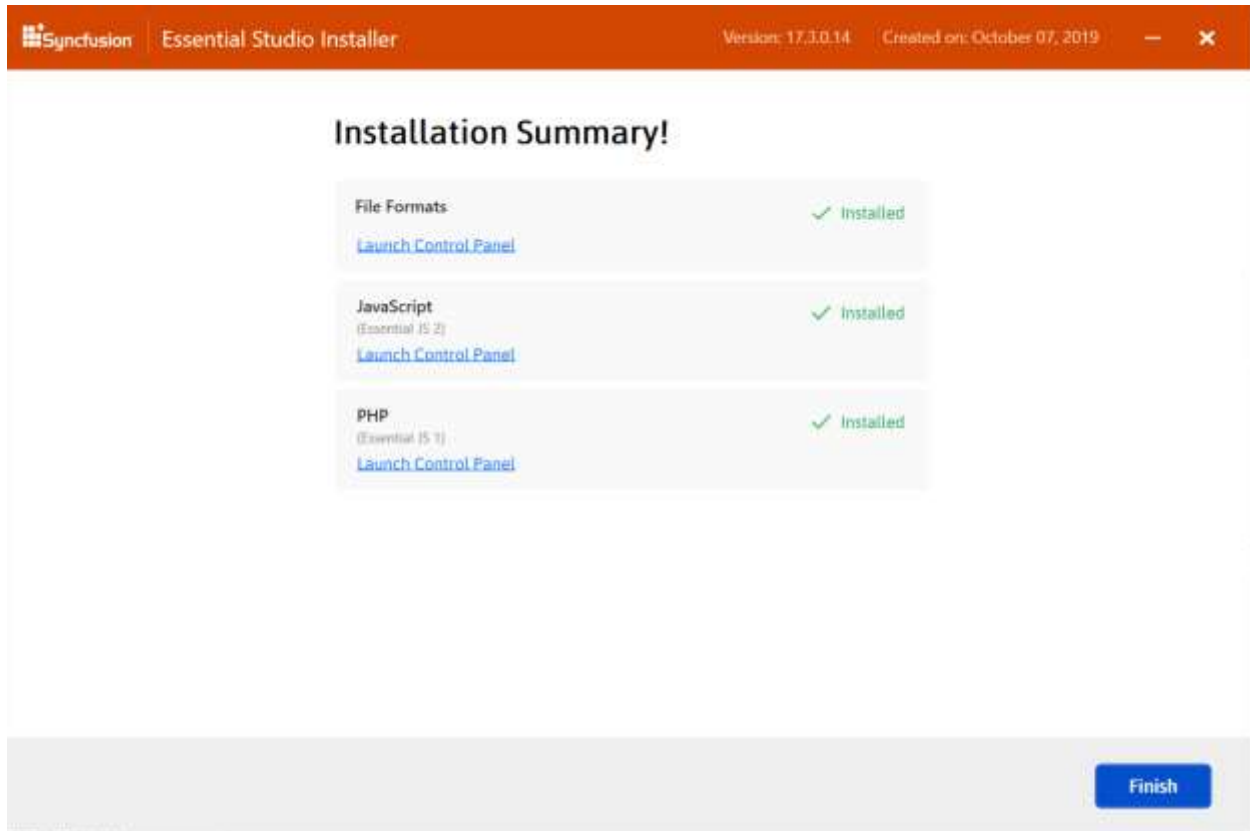
The screenshot shows the 'Essential Studio Installer' window. The title bar is orange and contains the Syncfusion logo, the text 'Essential Studio Installer', and version/creation information: 'Version: 17.3.0.14' and 'Created on: October 07, 2019'. The main content area has a light gray background and features the heading 'Just one more step!' followed by the instruction 'Please login to begin installation.' Below this are two input fields: 'Email' with a placeholder 'email id' and 'Password' with a placeholder 'password' and a toggle icon. At the bottom of the input fields are two links: 'Create an Account' and 'Forgot password?'. At the bottom right of the window are two buttons: 'Back' and 'Install'.

Information: The selected platforms will be installed based on your Syncfusion License (Trial or Licensed).

8. Download and Installation progress will be displayed.



9. Once the Installation is complete, **Summary** wizard will be displayed. Here you can check the list of platforms which are installed successfully and failed. Click Finish to exit the Summary wizard.



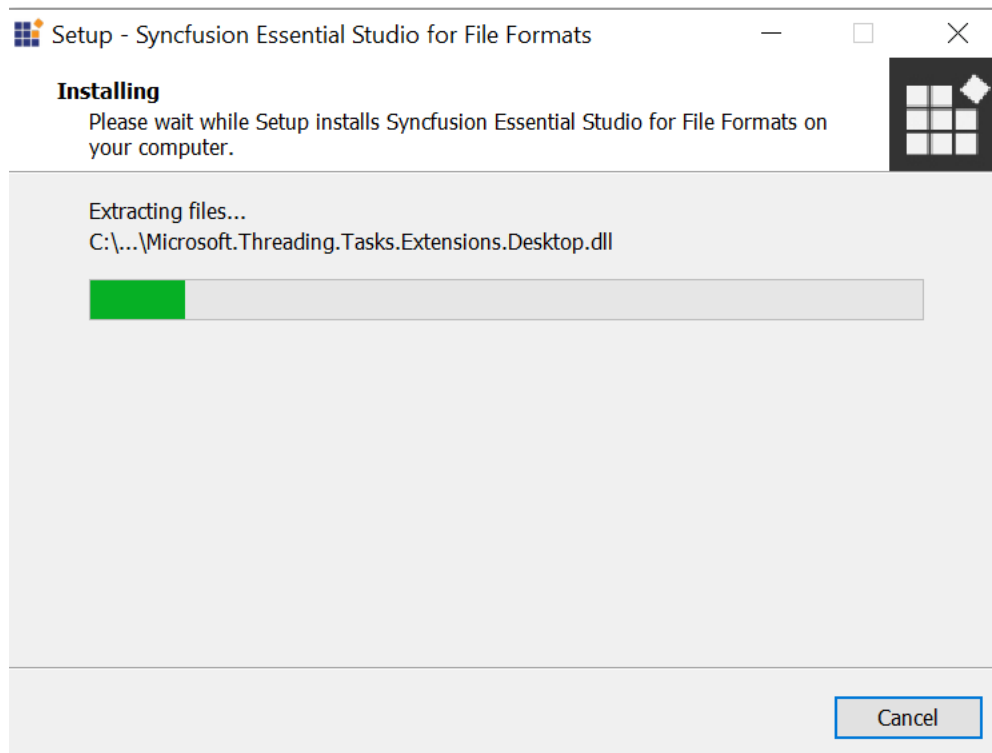
- Click **Launch Control Panel** to open the Syncfusion Control Panel.

Uninstallation

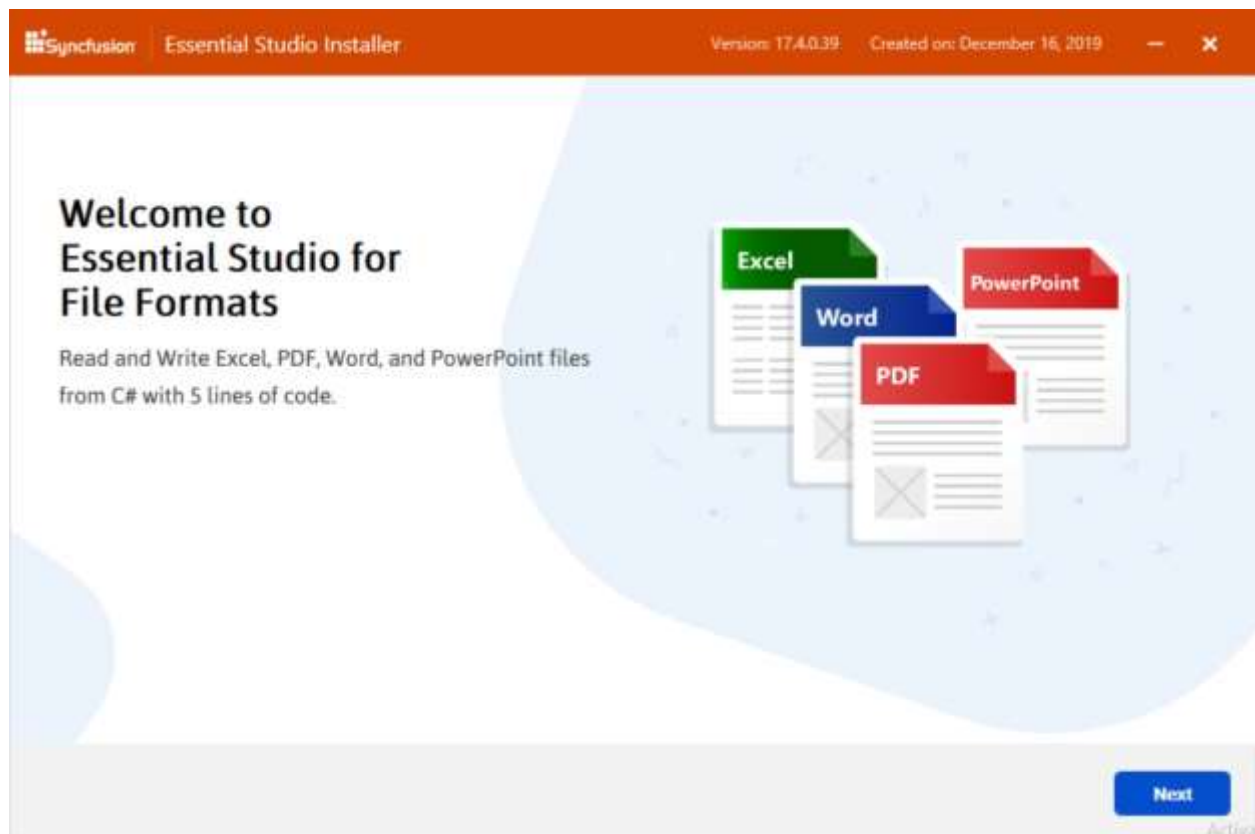
Till version 17.2, Syncfusion Web Installer had option for installation alone. Starting with the version 17.3 (2019 Vol 3), Syncfusion provides option for uninstalling the platforms of the same version from the Web Installer application itself. Select the list of the platforms to be uninstalled and Web Installer will uninstall those platforms one by one.

The following procedure illustrates how to uninstall Essential Studio Platform from Web Installer.

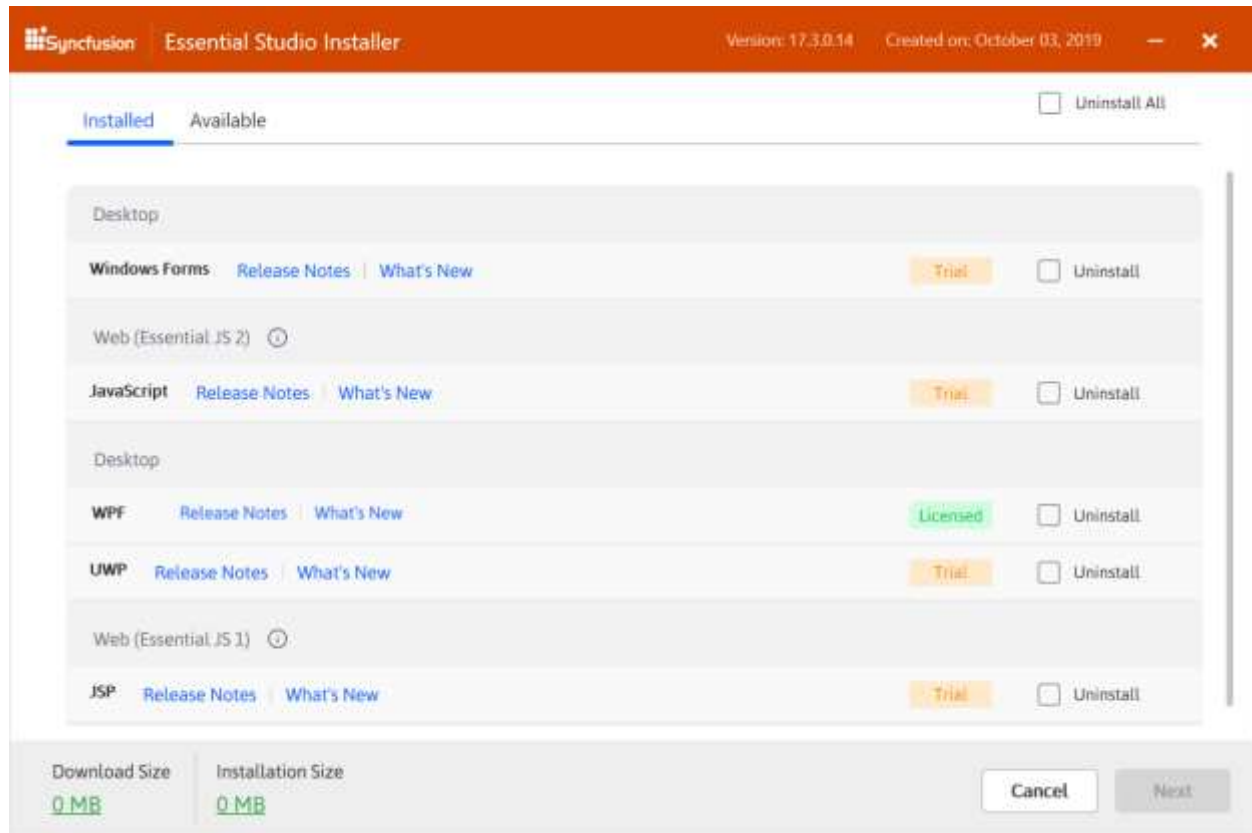
1. Double-click the Syncfusion Essential Studio Platform Online Installer file. The installer Wizard opens and extracts the package automatically.



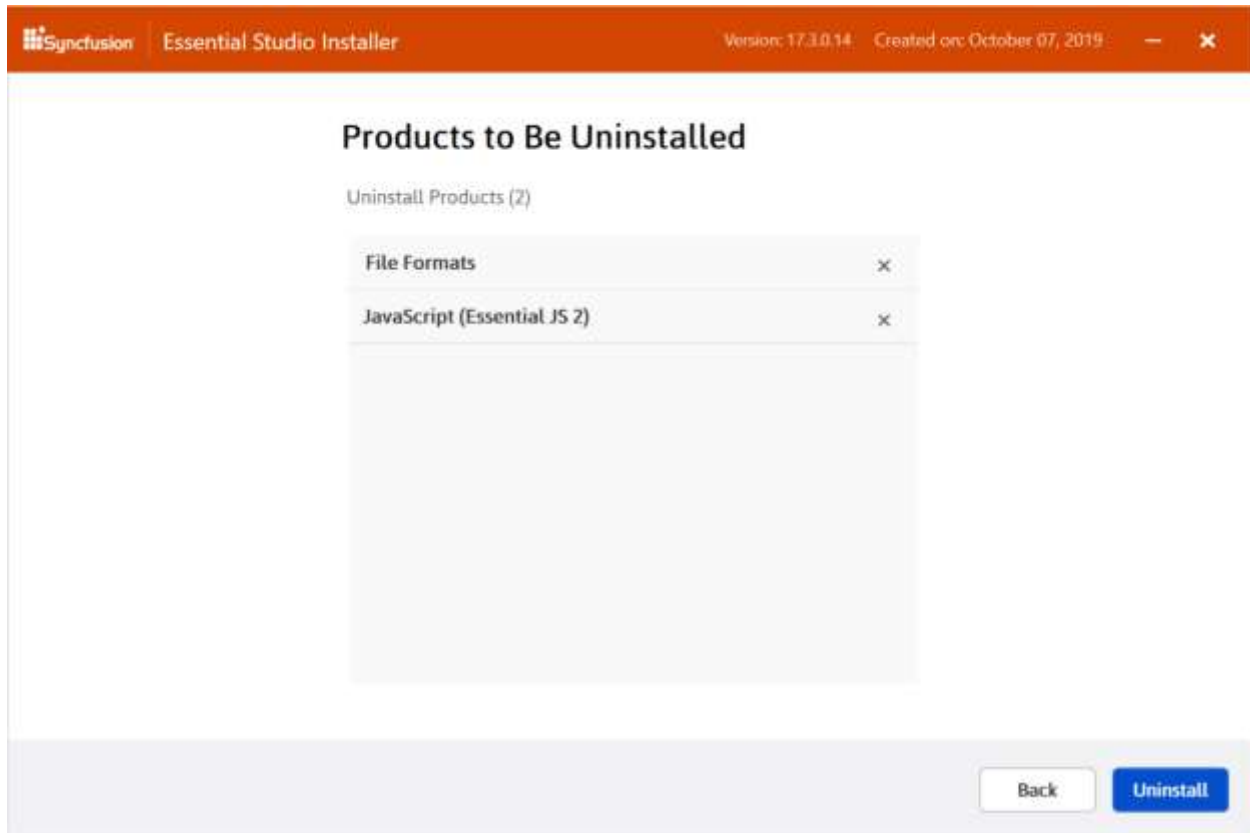
2. Welcome wizard of the Syncfusion Online Installer will be displayed. Click Next.



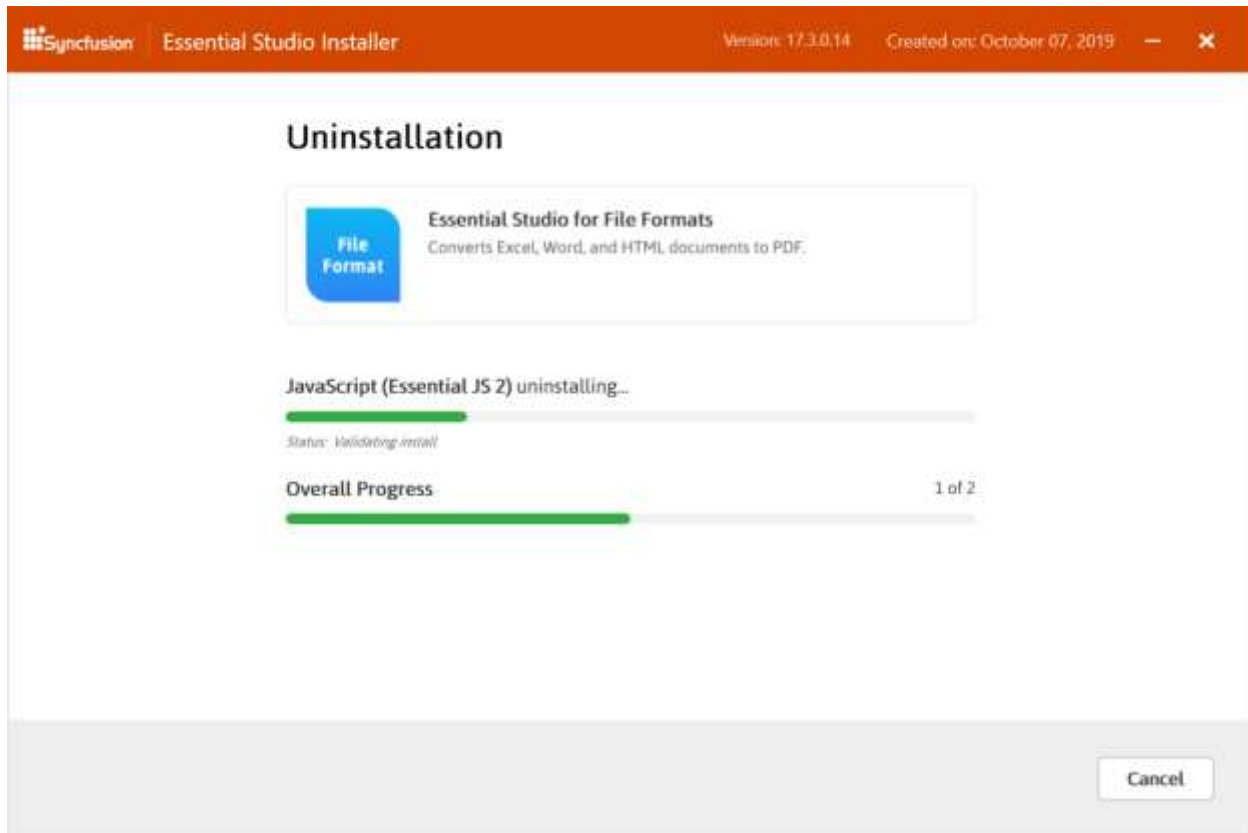
- Platform Selection wizard will be displayed. Select the platforms to be uninstalled from the **Installed** tab. Select **Uninstall All** checkbox to select all the platforms. Click Next.



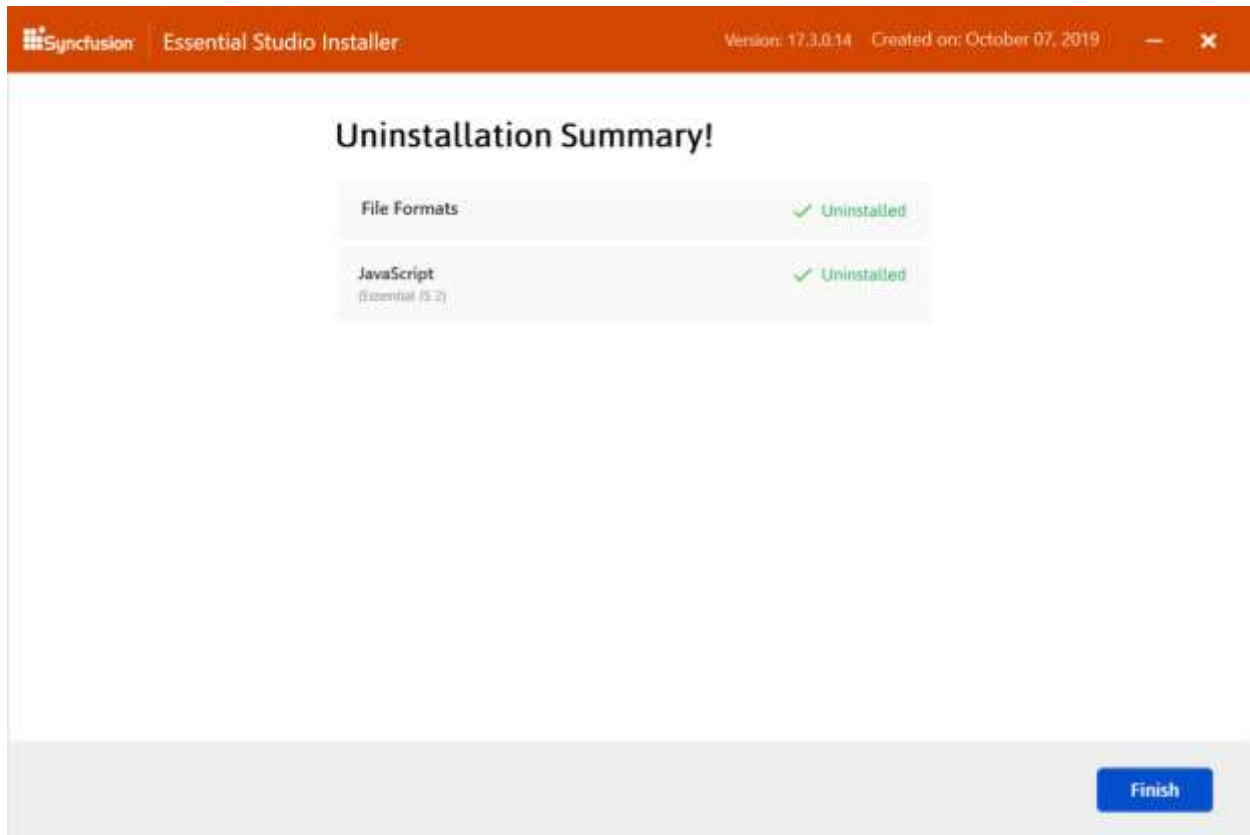
- Confirmation wizard will be displayed. Here you can view and modify the list of platforms that will be uninstalled. Click Uninstall.



5. Uninstallation progress will be displayed.



6. Once the Uninstallation is complete, **Summary** wizard will be displayed. Here you can check the list of platforms which are uninstalled successfully and failed. Click Finish to exit the Summary wizard.

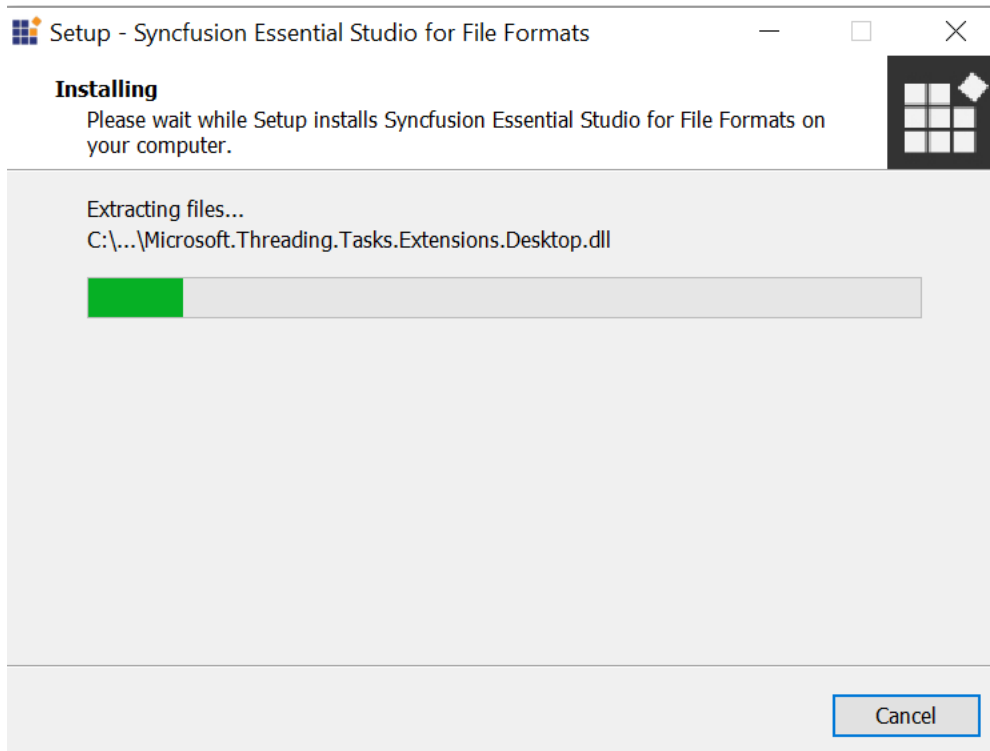


Installation and Uninstallation

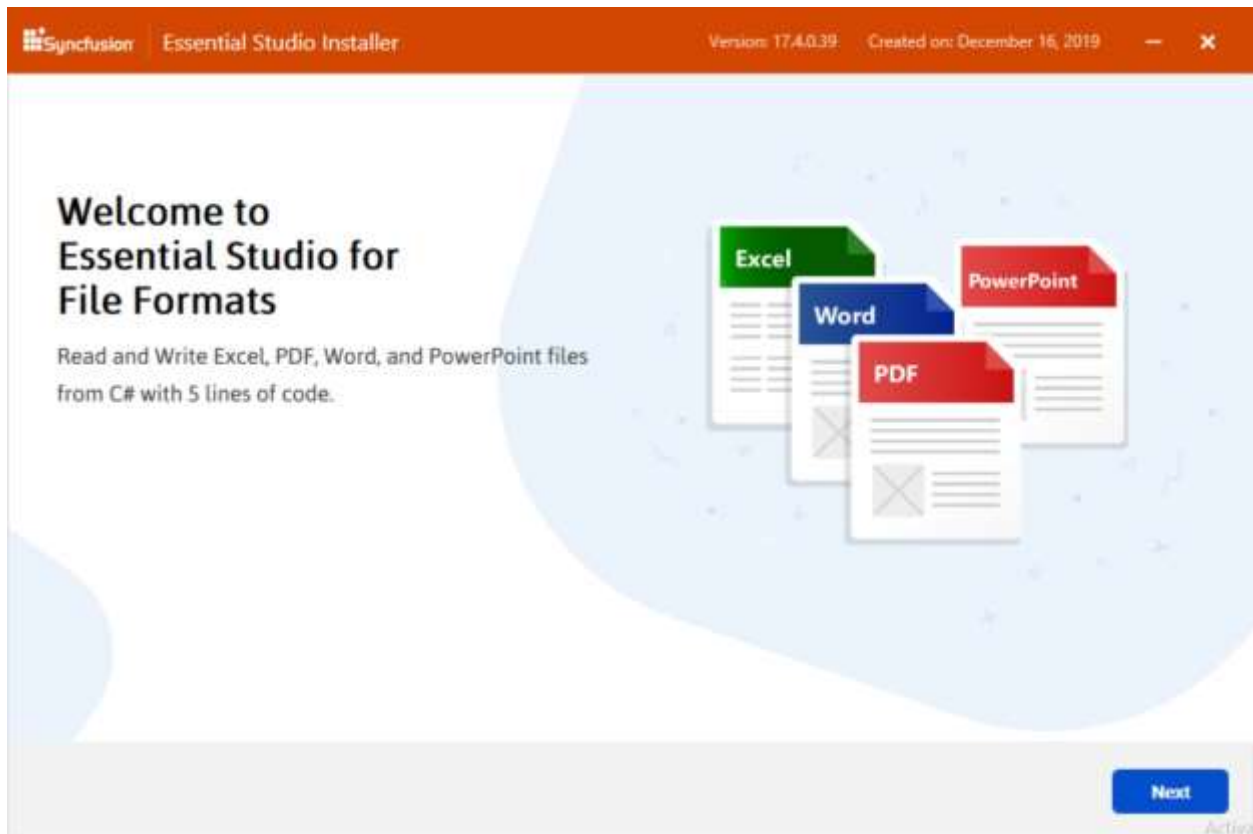
Till version 17.2, Syncfusion Web Installer had option for installation alone. Starting with the version 17.3 (2019 Vol 3), Syncfusion provides option for both install and uninstall the platforms of the same version from the Web Installer application itself.

The following procedure illustrates how to install/uninstall Essential Studio Platform from Web Installer.

1. Double-click the Syncfusion Essential Studio Platform Online Installer file. The installer Wizard opens and extracts the package automatically.

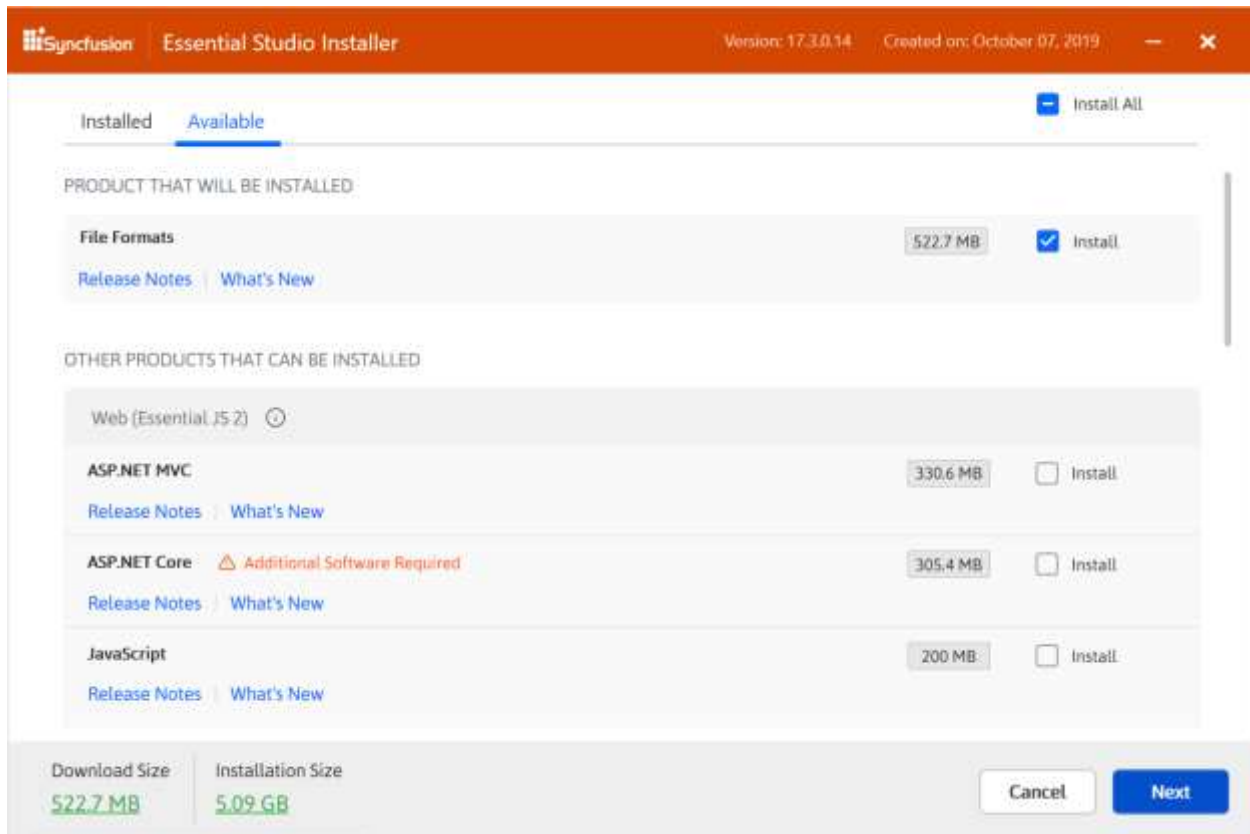


2. Welcome wizard of the Syncfusion Online Installer will be displayed. Click Next.

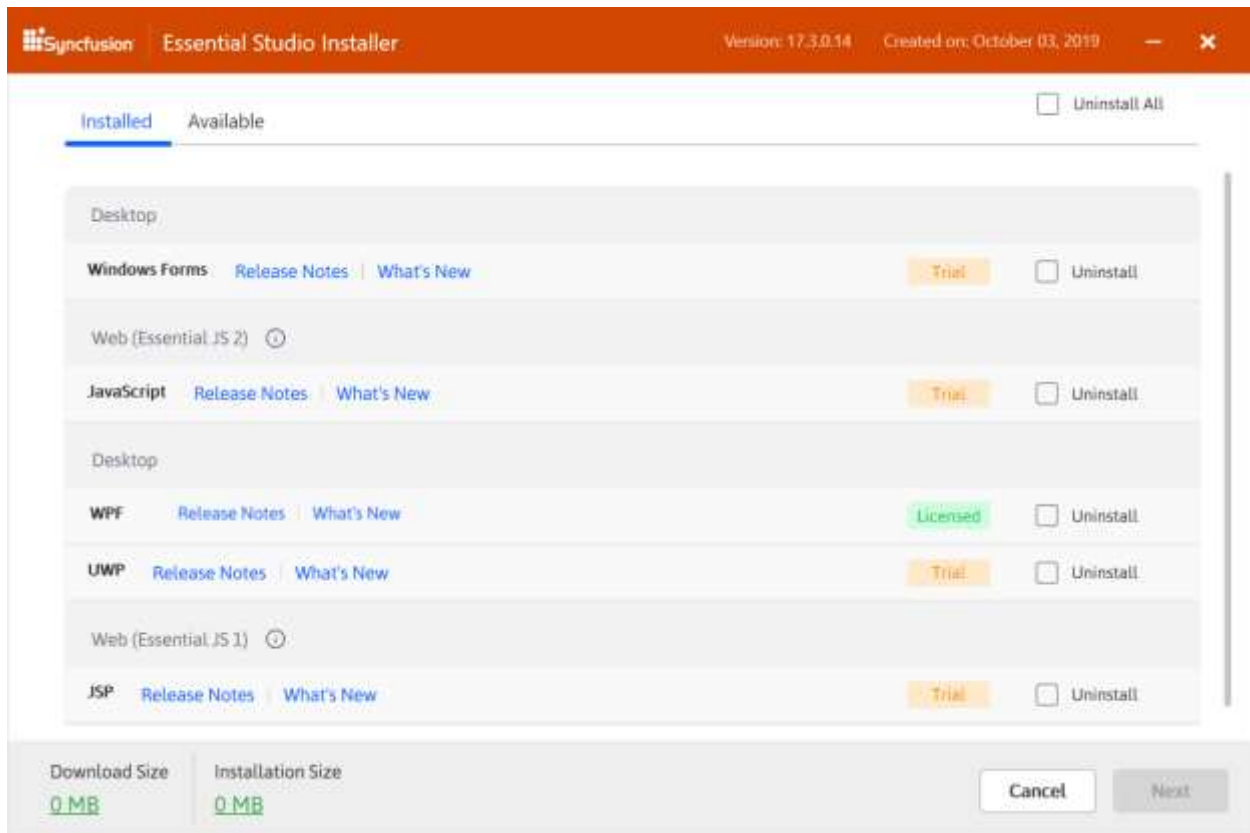


- Platform Selection wizard will be displayed. Select the platforms to be installed from the **Available** tab and platforms to be uninstalled from the **Installed** tab. Click Next.

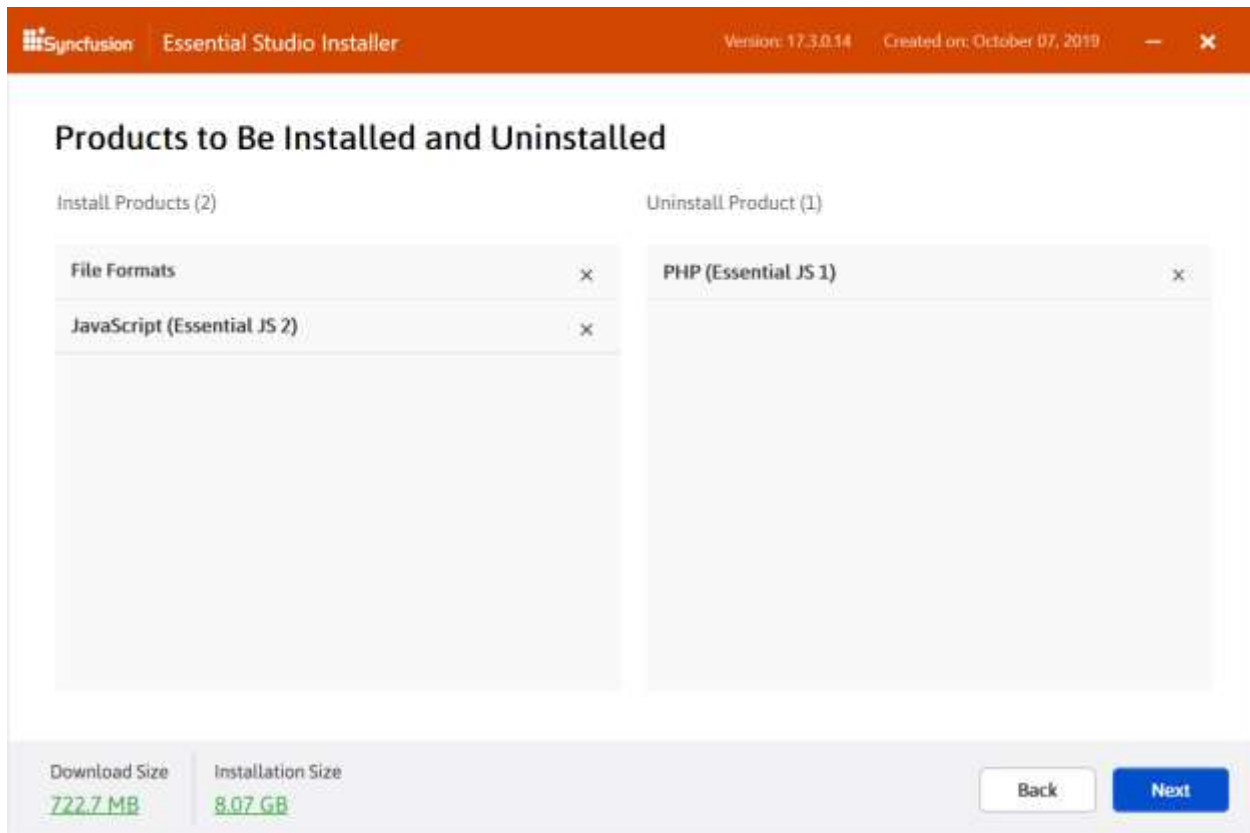
Available



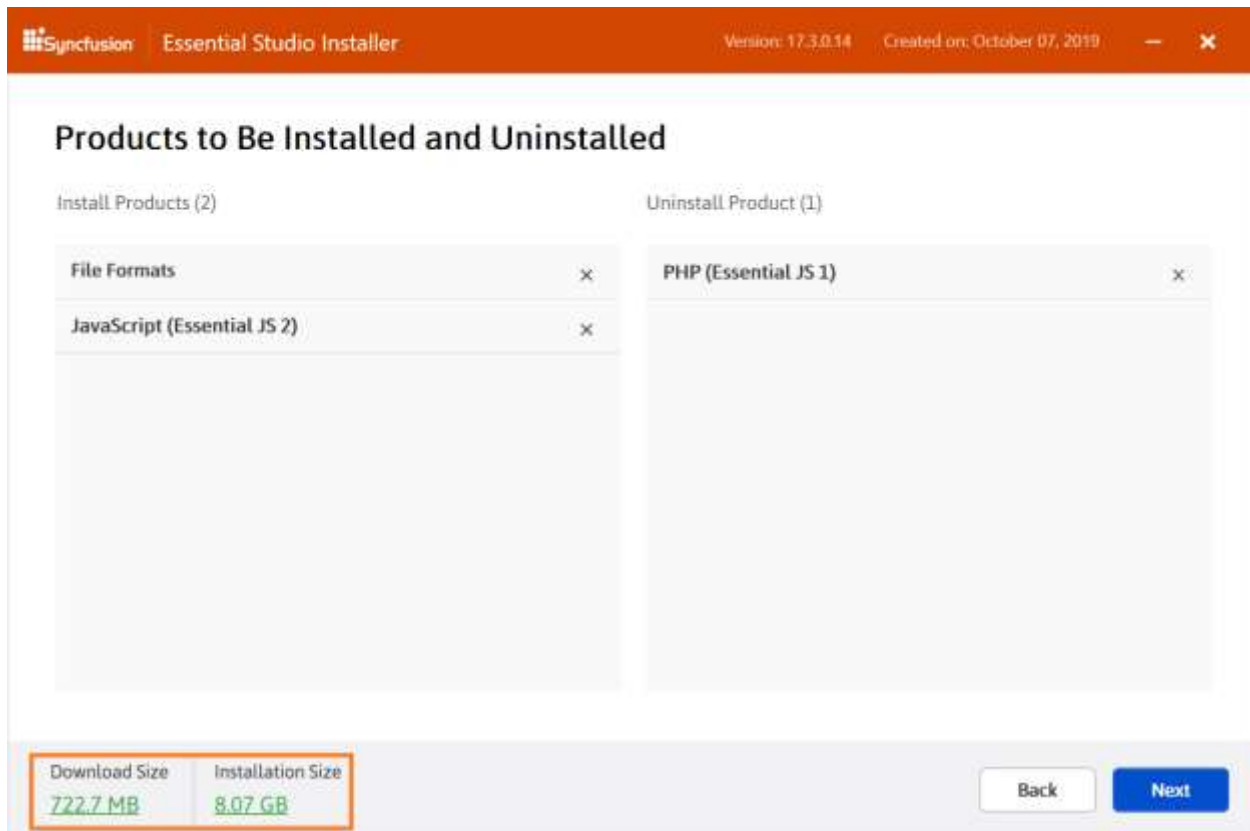
Installed



4. Confirmation wizard will be displayed. Here you can view and modify the list of platforms that will be installed/uninstalled.



Note: You can check the Estimated size of the Download and Installation by clicking the **Download Size and Installation Size** link.



5. Configuration wizard will be displayed. Here you can change the Download, Install and samples location. Also, you can change the Additional settings by platform basis. To install using the default configuration, click Next.

Configuration

Download Location
 C:\ProgramData\Syncfusion\17.3.0.14\Downloads\ [Browse](#)

Installation Location
 C:\Program Files (x86)\Syncfusion\Essential Studio\ [Browse](#)

Demos Location
 C:\Users\Public\Documents\Syncfusion\ [Browse](#)

☒ I Agree to the [License Terms](#) and [Privacy Policy](#)

Additional Settings

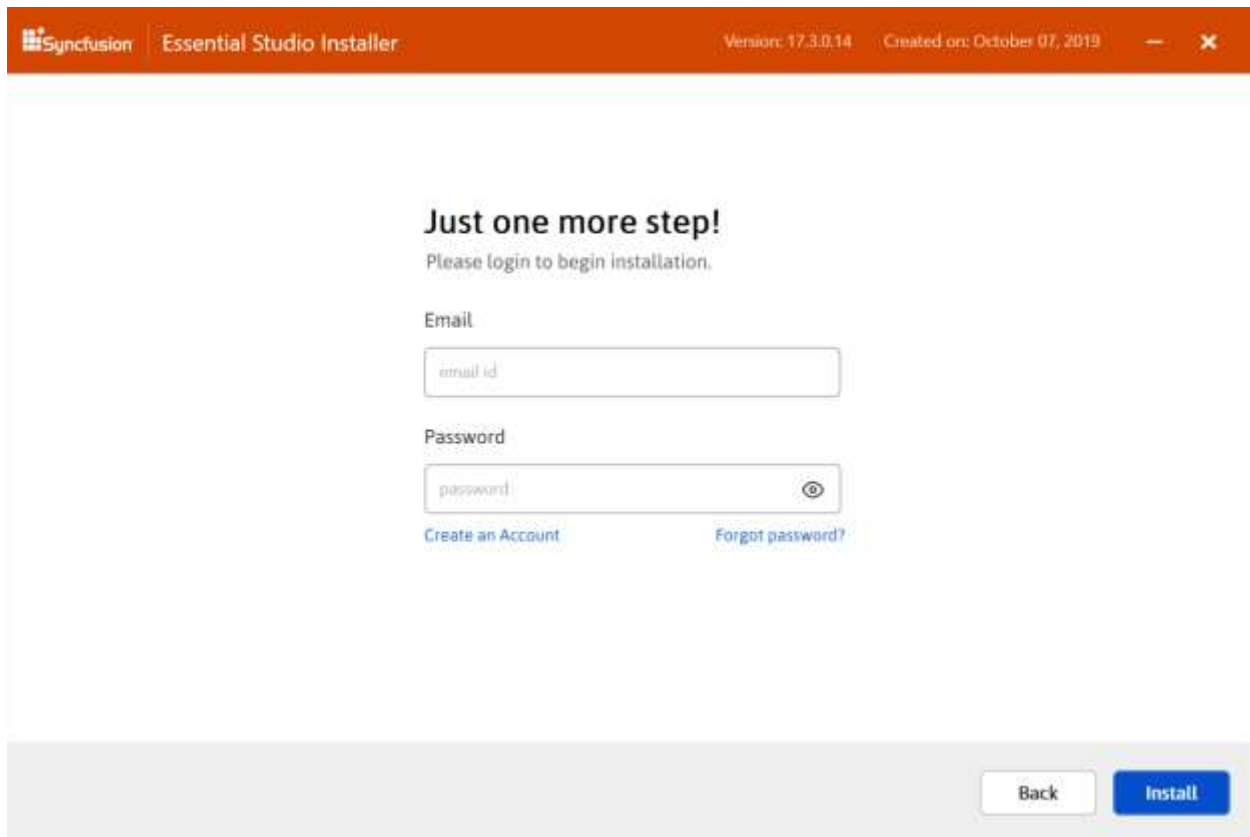
- ☒ Install Demos
 - ☒ File Formats
 - ☒ JavaScript (Essential JS 2)
 - ☒ PHP (Essential JS 1)
- ☒ Register Syncfusion assemblies in GAC
 - ☒ File Formats
- ☒ Configure Syncfusion Controls in Visual Studio
 - ☒ File Formats
- ☒ Create Desktop Shortcut(s)
 - ☒ File Formats
 - ☒ JavaScript (Essential JS 2)
 - ☒ PHP (Essential JS 1)

Download Size	Installation Size
740.5 MB	8.15 GB

[Back](#) [Next](#)

Information: From version 17.3 (2019 Vol 3), Syncfusion provides option to provide custom download location.

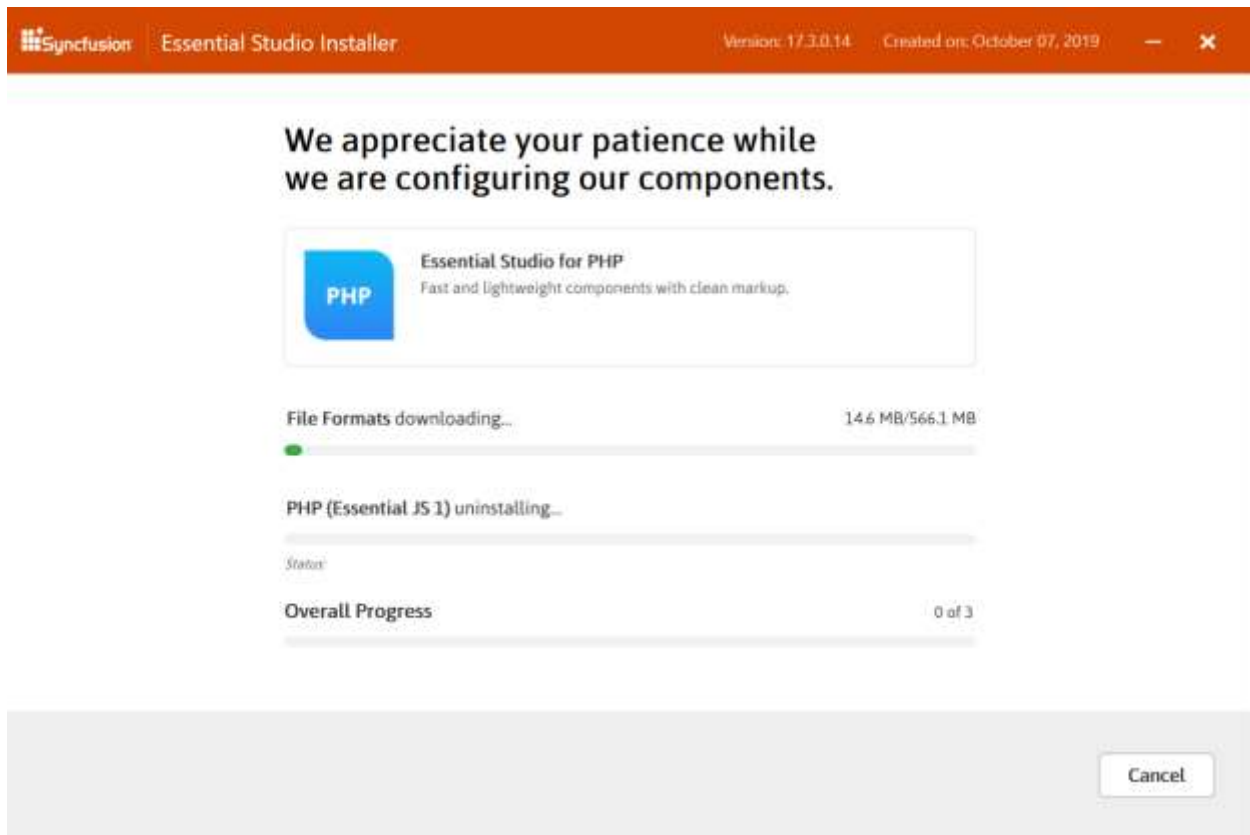
- After reading the License Terms and Conditions, check the **I agree to the License Terms and Privacy Policy** check box. Click Next.
- Login wizard will be displayed. You should enter your Syncfusion Direct-Trac login credentials. If you don't have Syncfusion Direct-Trac login credentials, then you can click on **Create an Account**. Else if you forgot your password, click on **Forgot Password** to create new password. Click Install.



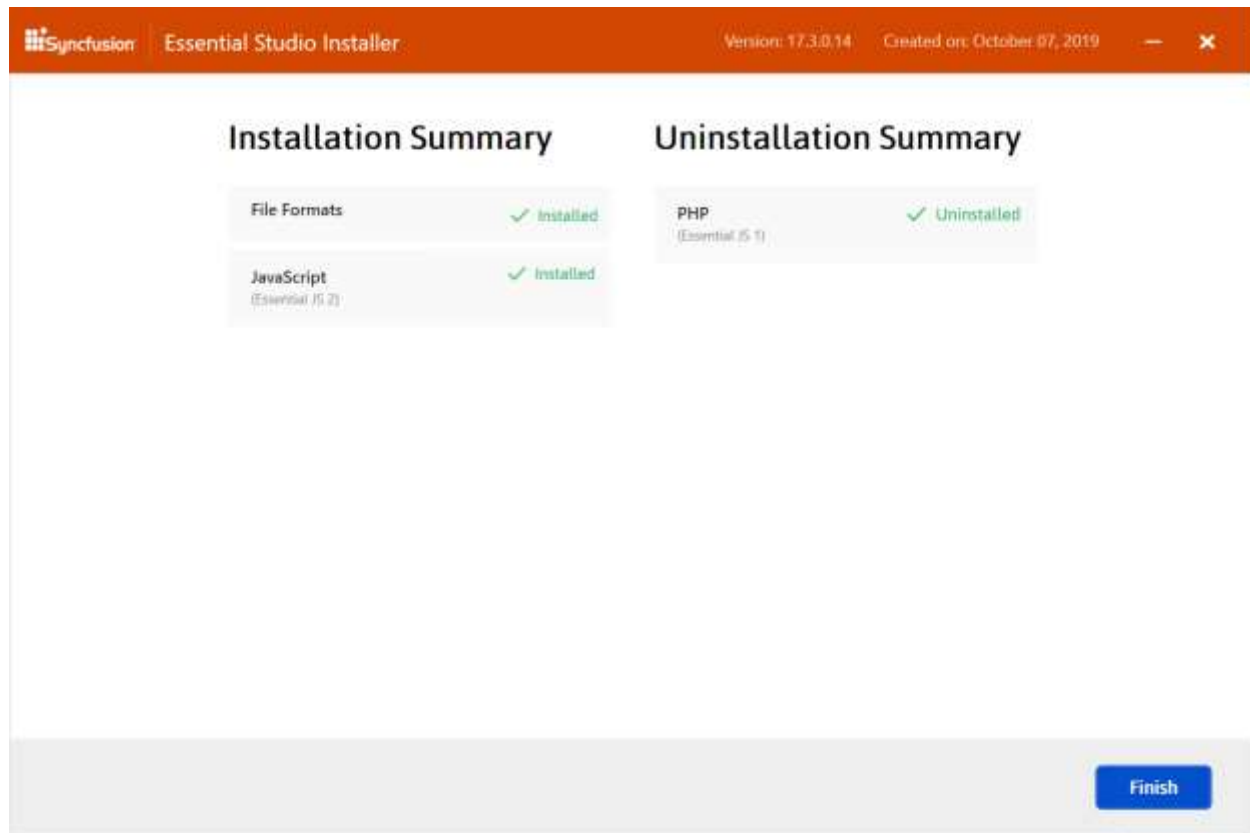
The screenshot shows the 'Essential Studio Installer' window. The title bar is orange and contains the Syncfusion logo, the text 'Essential Studio Installer', and version/creation information: 'Version: 17.3.0.14' and 'Created on: October 07, 2019'. The main content area is white and features the heading 'Just one more step!' followed by the instruction 'Please login to begin installation.' Below this are two input fields: 'Email' with a placeholder 'email id' and 'Password' with a placeholder 'password' and a toggle icon. At the bottom of the login section are two links: 'Create an Account' and 'Forgot password?'. At the very bottom of the window are two buttons: 'Back' and 'Install'.

Information: The selected platforms will be installed based on your Syncfusion License (Trial or Licensed).

8. Download, Installation and Uninstallation progress will be displayed.



9. Once the Installation is complete, **Summary** wizard will be displayed. Here you can check the list of platforms which are **installed/uninstalled** successfully and failed. Click Finish to exit the Summary wizard.



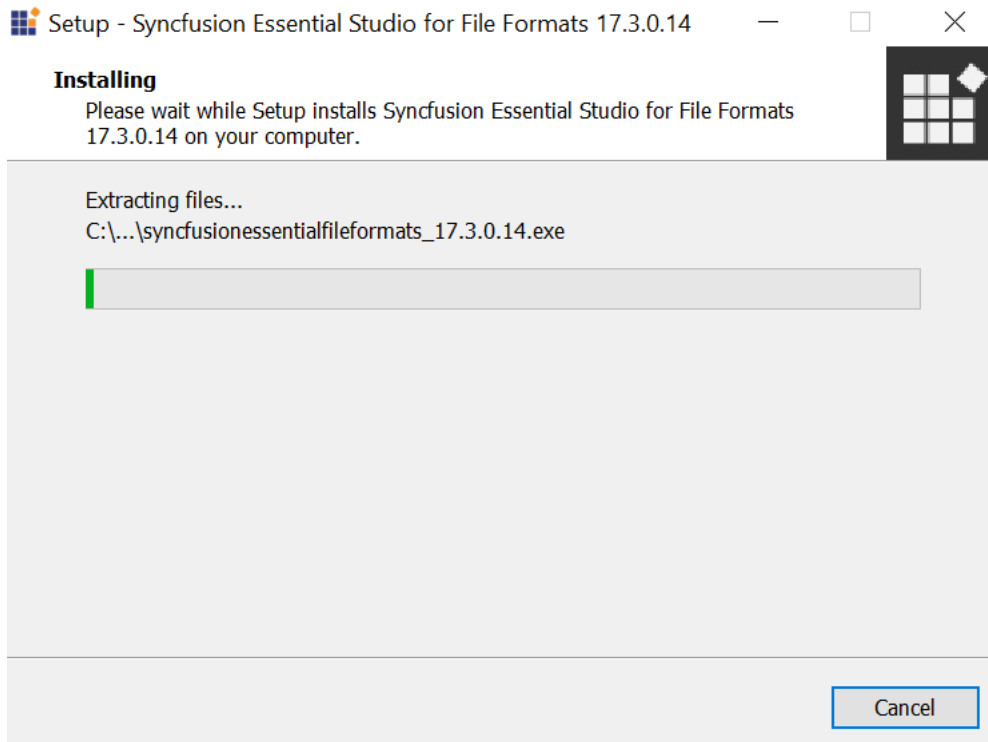
- Click **Launch Control Panel** to open the Syncfusion Control Panel.

Installation using Offline Installer

Installing with UI

The following procedure illustrates how to install Essential Studio File Formats platform.

1. Close all the running Visual Studio instances.
2. Double-click the Syncfusion File Formats platform installer file. The installer Wizard opens and extracts the package automatically.



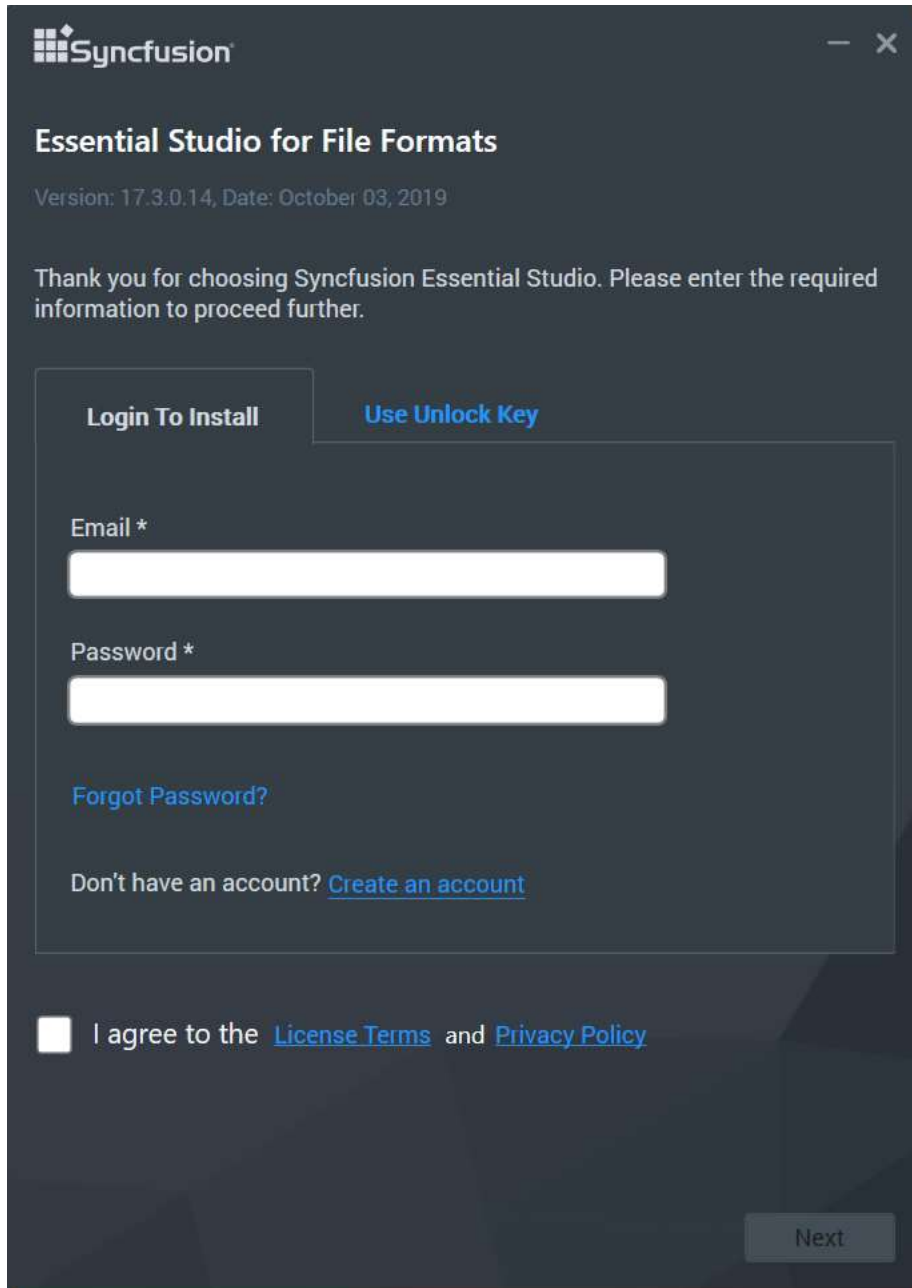
Note: The installer wizard extracts the syncfusionessentialfileformats_(version).exe dialog, displaying the unzip operation of the package.

3. You have two options to unlock the Syncfusion installer:

- *Login To Install*
- *Use Unlock Key*

Login To Install

You should enter your Syncfusion Direct-Trac login credentials. If you don't have Syncfusion Direct-Trac login credentials, then you can click on Sign Up to create a new account. Else if you forgot your password, click on Reset Password to create a new password. Here Email address and Password is validated and the Platform Selection window opens.



Syncfusion

Essential Studio for File Formats

Version: 17.3.0.14, Date: October 03, 2019

Thank you for choosing Syncfusion Essential Studio. Please enter the required information to proceed further.

Login To Install **Use Unlock Key**

Email *

Password *

[Forgot Password?](#)

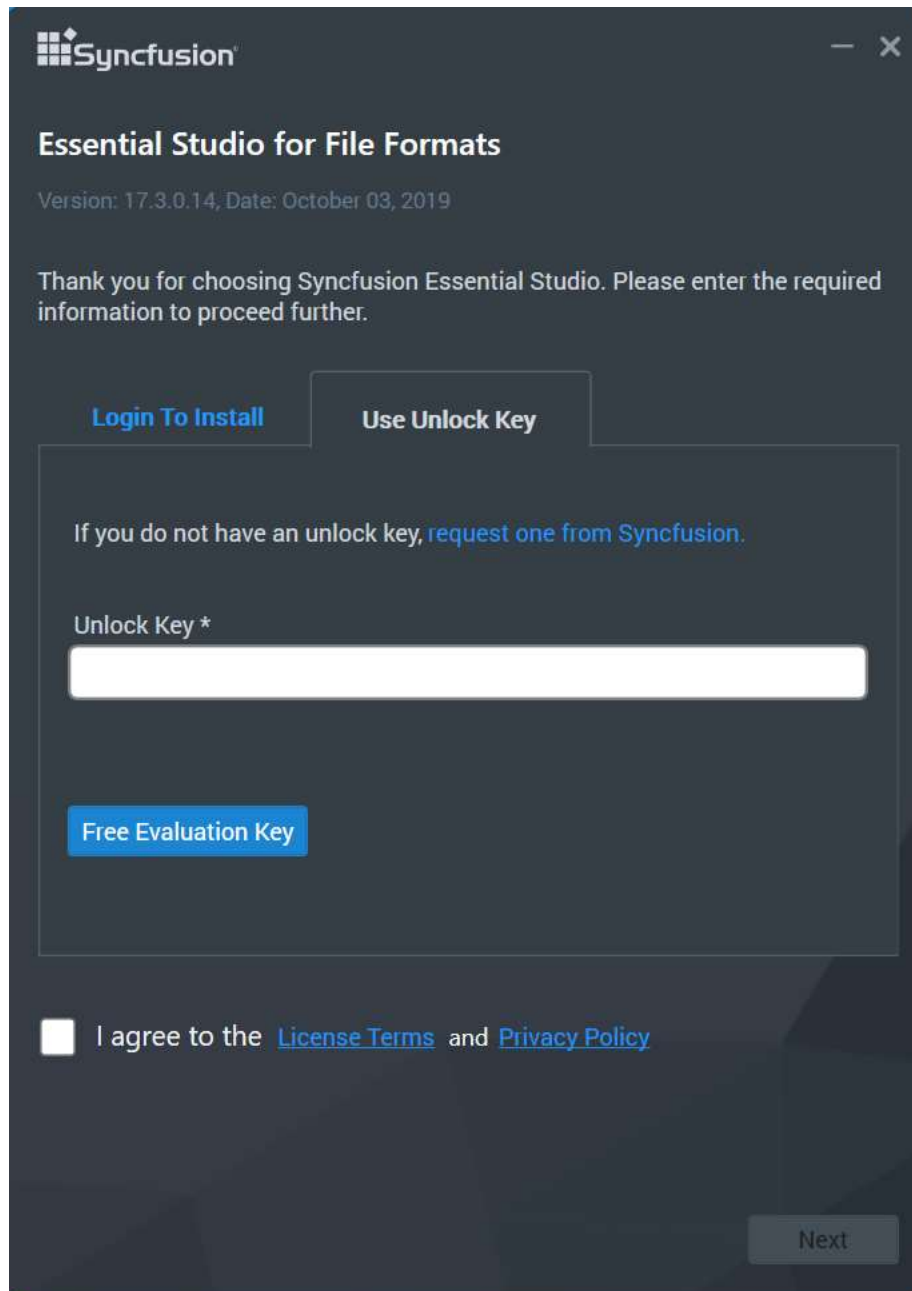
Don't have an account? [Create an account](#)

☐ I agree to the [License Terms](#) and [Privacy Policy](#)

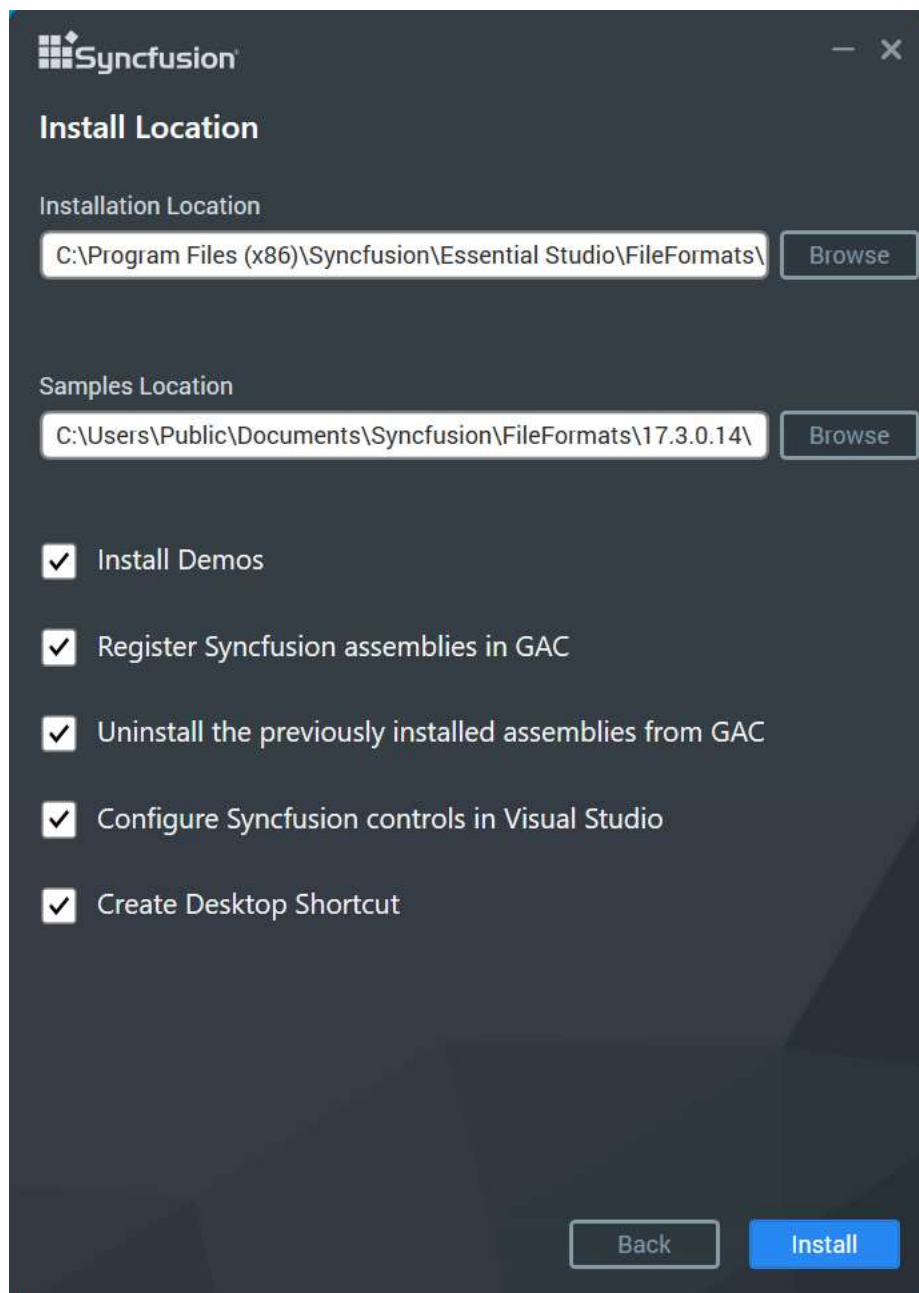
Next

Use Unlock Key

You should use the Syncfusion License/Trial key. Trial key is valid for 30 days and the installer won't accept the expired trial key. Licensed customer can generate key from [here](#).



4. After reading the License Terms and Conditions, check the **I agree to the License Terms and Conditions** check box.
5. Click Next. Select the Installation, Samples Folder and Advanced Options screen opens. To install in the displayed default location, click Install



Note: From the 2018 Volume 2 release, Syncfusion has changed the install and samples location

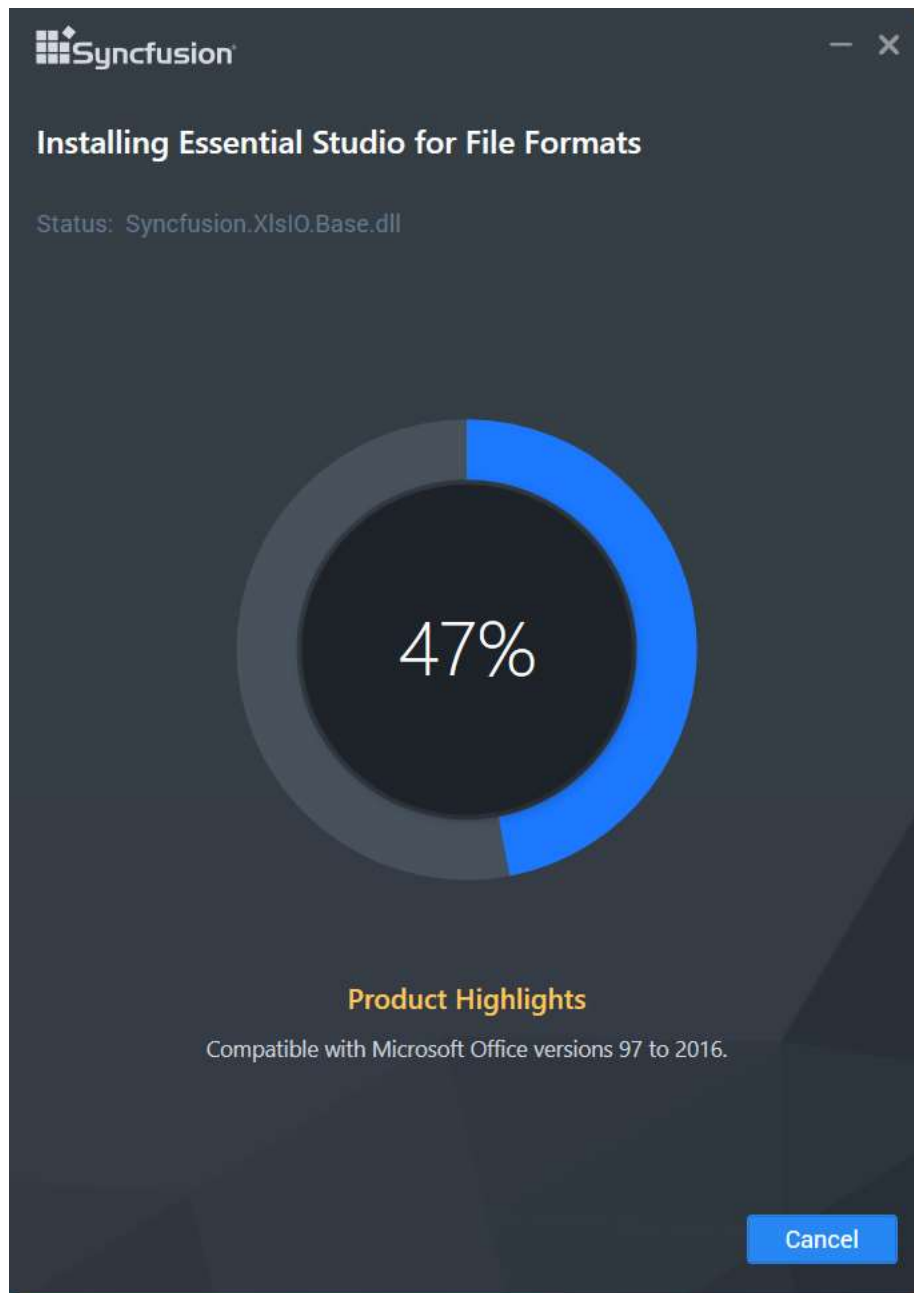
Default Install location: {ProgramFilesFolder}\Syncfusion\{Platform}\{version}

Default Samples location: C:\Users\Public\Documents\Syncfusion\{platform}\{version}

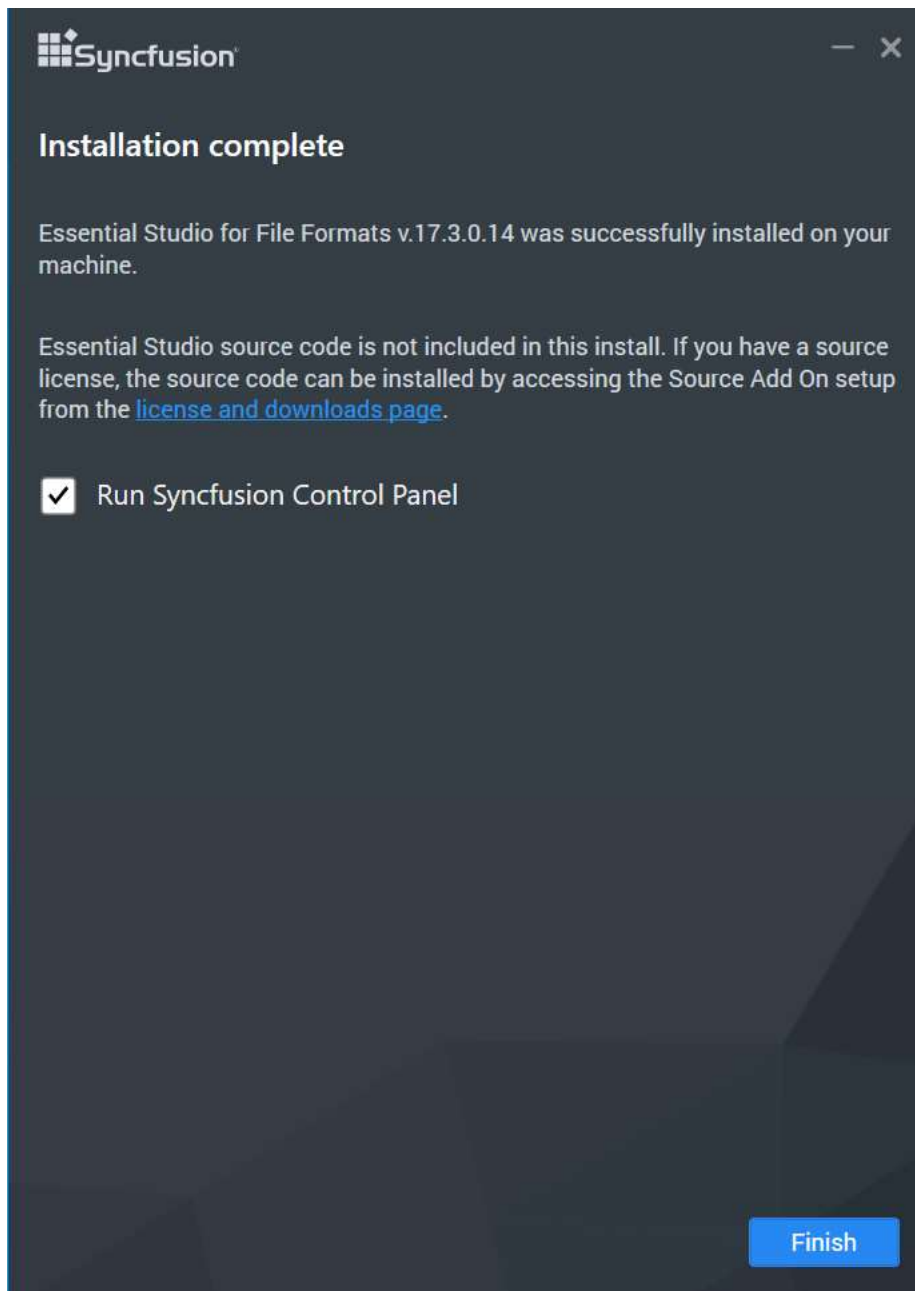
However, you can change the locations by clicking browse button.

- Select the **Install Syncfusion Samples** check box to install Syncfusion samples, or leave the check box clear, when you do not want to install Syncfusion samples.
- Select the **Register Syncfusion Assemblies in GAC** check box to install the latest Syncfusion assemblies in GAC, or clear this check box when you do not want to install the latest assemblies in GAC.

- Select the **Uninstall the previously installed Syncfusion assemblies from GAC** check box to uninstall the previously installed Syncfusion assemblies from GAC, or clear this check box to maintain the previously installed assemblies.
 - Select the **Configure Syncfusion controls in Visual Studio Toolbox** check box to configure the Syncfusion controls in the Visual Studio toolbox, or clear this check box when you do not want to configure the Syncfusion controls in the Visual Studio toolbox during installation. Note that you must also select the Register Syncfusion assemblies in GAC check box when you select this check box.
 - Select the **Install Syncfusion Extensions** checkbox to configure the Syncfusion Extensions in Visual Studio or clear this check box when you do not want to configure the Syncfusion Extensions in Visual Studio.
6. Click Install.



7. The Completed screen is displayed once the File Formats platform is installed.



8. Select the **Run Syncfusion Control Panel** check box to launch the Syncfusion Control Panel after installing.
9. Click Finish. Syncfusion File Formats platform is installed in your system and the Syncfusion Essential Studio [Syncfusion Control Panel](#) is launched automatically.

Installing in silent mode

The Syncfusion Essential Studio Platform Installer supports installing/uninstalling through Command Line. The following sections illustrate this ability.

Command Line Installation

Follow the steps below to install through Command Line in Silent mode.

1. Double-click the Syncfusion Essential Studio platform installer file. The installer Wizard opens and extracts the package automatically.
2. The syncfusionessentialfileformats_(version).exe file is extracted into the Temp folder.
3. Run %temp%. The Temp folder will open. The syncfusionessentialfileformats_(version).exe file is available in one of the folders.
4. Copy the extracted syncfusionessentialfileformats_(version).exe file in local drive.
5. Cancel the Wizard.
6. Open the Command Prompt in administrator mode and pass the following arguments.

Arguments: "Installer file path\SyncfusionEssentialStudio(platform)_(version).exe" /Install silent /PIDKEY:"(product unlock key)" [/log "{Log file path}"] [/InstallPath:{Location to install}] [/InstallSamples:{true/false}] [/InstallAssemblies:{true/false}] [/UninstallExistAssemblies:{true/false}] [/InstallToolbox:{true/false}]

Note: [...] – Arguments inside the square brackets are optional.

Example: "D:\Temp\syncfusionessentialfileformatsx.x.x.x.exe" /Install silent /PIDKEY:"product unlock key" /log "C:\Temp\EssentialStudioPlatform.log" /InstallPath:C:\Syncfusion\x.x.x.x /InstallSamples:true /InstallAssemblies:true /UninstallExistAssemblies:true /InstallToolbox:true

7. Essential Studio for File Formats is installed.

Note: x.x.x.x needs to be replaced with the Essential Studio version and the Product Unlock Key needs to be replaced with the Unlock Key for that version.

Command Line Uninstallation

Syncfusion Essential Studio supports uninstalling through Command Line in Silent mode. The following steps illustrate this.

1. When you do not have the extracted installer (syncfusionessentialfileformats_(version).exe) then follow the steps from 2 to 7.
2. Double-click the Syncfusion Essential Studio platform installer file. The installer Wizard opens and extracts the package automatically.
3. The syncfusionessentialfileformats_(version).exe file is extracted into the Temp folder.
4. Run %temp%. The Temp folder will open. The syncfusionessentialfileformats_(version).exe file is available in one of the folders.
5. Copy the syncfusionessentialfileformats_(version).exe file in local drive.
6. Cancel the Wizard.
7. Open the Command Prompt in administrator mode and pass the following arguments.

Arguments: "Copied installer file path\syncfusionessentialfileformats_(version).exe" /uninstall silent

Example: "D:\Temp\syncfusionessentialfileformats_x.x.x.x.exe" /uninstall silent

8. Essential Studio for File Formats is uninstalled.

Note: x.x.x.x needs to be replaced with the Essential Studio version installed in your machine.

Installation FAQ

Refer [this](#) topic for more information regarding the issues related to installation.

Upgrade from one version to another version.

You can upgrade to the latest version by downloading and installing the platforms you require from [this](#) link.

Upgrade from major version to service pack version

Syncfusion provides a new Volume release once in every three months which has exciting new features. For that Volume release, there may be one or two Service Pack releases. The issues in the Volume release will be addressed in the Service Pack releases. You can download and install the latest Service Pack installer [here](#).

It is not required to install the Volume release before installing the Service Pack release. As Volume and Service Packs releases works independently, you can directly install the latest Service Pack contains major issue fixes.

Upgrade from trial version to license version

To upgrade from trial version, there are two possible solutions.

- Uninstall the trial version and install the fully licensed build from the [License & Downloads](#) section of our website.
- Replace the currently used trial license key with a paid license key that can be generated from the License & Downloads section of our website.

NuGet Packages

[NuGet](#) can be used to automatically add files and references to your Visual Studio projects. You can use the Syncfusion NuGet packages without installing the Essential Studio or platform installation to development with the Syncfusion controls. From v16.2.0.46 (2018 Volume 2 Service Pack 1) onwards, all the Syncfusion components are available as NuGet packages at [nuget.org](#).

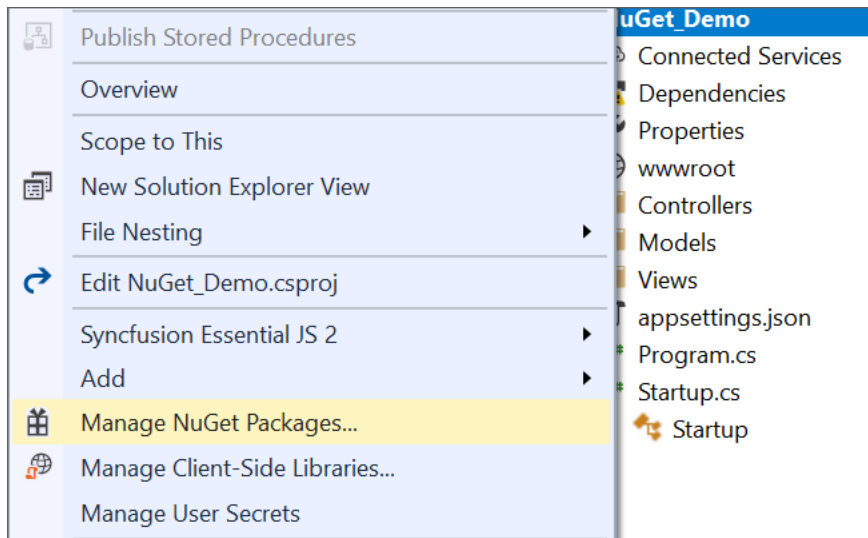
Note: The Syncfusion Xamarin NuGet packages are published in [NuGet.org](#) from v15.4.0.17. Starting from v17.1.0.32 (2018 Volume 1), Syncfusion will no longer publish NuGet packages at [nuget.syncfusion.com](#).

Installing NuGet Packages

Using NuGet Package Manager

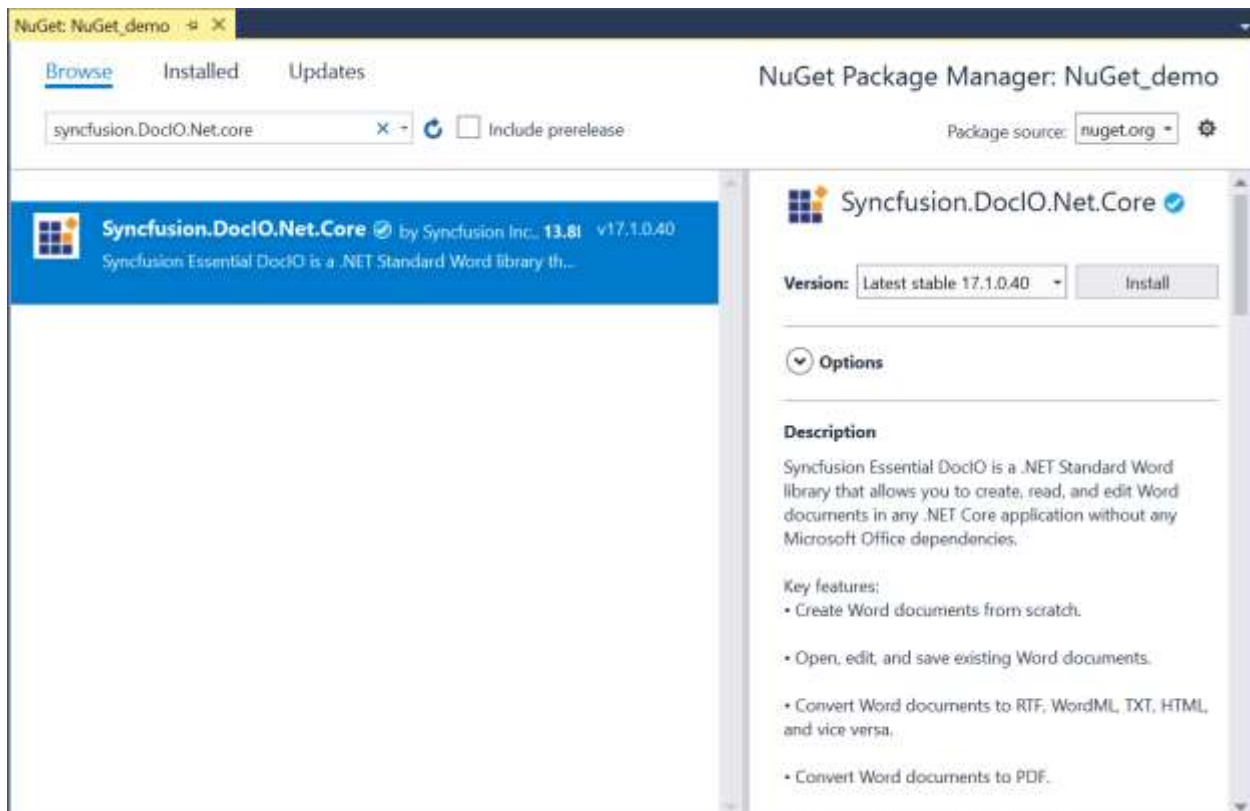
The NuGet Package Manager can be used to search and install NuGet packages in the Visual Studio solution or project:

1. Right-click the project or solution in the Solution Explorer tab, and choose **Manage NuGet Packages...**



Alternatively, click **Tools** menu, **NuGet Package Manager | Manage NuGet Packages for Solution...**

2. By default, the NuGet.org package is selected in the **Package source** drop-down. If NuGet.org is not configured in your Visual Studio, refer to the [Microsoft docs](#) to configure NuGet.org feed URL in your Visual Studio.



3. The SynCFusion NuGet Packages are listed the available package in the source feed URL. Search and install the required packages in your application, by clicking **Install** button.

Note: The Syncfusion NuGet packages are published in public [NuGet.org](https://www.nuget.org/packages/Syncfusion.DocIO) from v16.2.0.46. So, If you need to Install earlier version of 16.2.0.46 Syncfusion NuGet packages, [configure Syncfusion private feed URL](#).

Using Package Manager Console

To reference the Syncfusion component using the Package Manager Console as NuGet packages, follow the below steps.

1. On the **Tools** menu, select **NuGet Package Manager**, and then **Package Manager Console**. 2. Run the following NuGet installation commands.

DocIO

C#

```
using Syncfusion.DocIO;  
using Syncfusion.DocIO.DLS;
```

VB.NET

```
Imports Syncfusion.DocIO  
Imports Syncfusion.DocIO.DLS
```

UWP

```
using Syncfusion.DocIO;  
using Syncfusion.DocIO.DLS;
```

ASP.NET CORE

```
using Syncfusion.DocIO;  
using Syncfusion.DocIO.DLS;
```

XAMARIN

```
using Syncfusion.DocIO;  
using Syncfusion.DocIO.DLS;
```

Creating a new Word document with few lines of code

The following code example explains how to create a new Word document with few lines of code

C#

```
//Creates an instance of WordDocument Instance (Empty Word Document)  
WordDocument document = new WordDocument();  
//Add a section & a paragraph in the empty document  
document.EnsureMinimal();  
//Append text to the last paragraph of the document  
document.LastParagraph.AppendText("Hello World");  
//Save and close the Word document  
document.Save("Result.docx");  
document.Close();
```

VB.NET

```
'Creates an instance of WordDocument Instance (Empty Word Document)
Dim document As New WordDocument()
'Add a section & a paragraph in the empty document
document.EnsureMinimal()
'Append text to the last paragraph of the document
document.LastParagraph.AppendText("Hello World")
'Save and close the Word document
document.Save("Result.docx")
document.Close()
```

UWP

```
//Creates an instance of WordDocument Instance (Empty Word Document)
WordDocument document = new WordDocument();
//Add a section & a paragraph in the empty document
document.EnsureMinimal();
//Append text to the last paragraph of the document
document.LastParagraph.AppendText("Hello World");
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new instance of WordDocument (Empty Word Document)
WordDocument document = new WordDocument();
//Adds a section and a paragraph to the document
document.EnsureMinimal();
//Appends text to the last paragraph of the document
document.LastParagraph.AppendText("Hello World");
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
stream.Position = 0;
document.Close();
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates an instance of WordDocument Instance (Empty Word Document)
WordDocument document = new WordDocument();
//Add a section & a paragraph in the empty document
document.EnsureMinimal();
//Append text to the last paragraph of the document
document.LastParagraph.AppendText("Hello World");
//Saves the Word document to MemoryStream
```

```

MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Creating a new Word document from scratch with basic elements

An entire Word document is represented by an instance of `WordDocument` and it is root element of DocIO's DOM. Word document contains a collection of sections. A Word document must contain at least one section.

A section represents group of paragraphs, tables etc., that have a specific set of properties used to define the pages, number of columns, headers and footers and so on that decides how the text appears. A section should contain at least one paragraph in this body.

The following code example explains how to add a section into a `WordDocument` instance.

C#

```

//Creates an instance of WordDocument Instance (Empty Word Document)
WordDocument document = new WordDocument();
//Adds a new section into the Word document
IWSection section = document.AddSection();
//Specifies the page margins
section.PageSetup.Margins.All = 50f;

```

VB.NET

```

'Creates an instance of WordDocument Instance (Empty Word Document)
Dim document As New WordDocument()
'Adds a new section into the Word document
Dim section As IWSection = document.AddSection()
'Specifies the page margins
section.PageSetup.Margins.All = 50.0F

```

UWP

```

//Creates an instance of WordDocument Instance (Empty Word Document)
WordDocument document = new WordDocument();
//Adds a new section into the Word document
IWSection section = document.AddSection();
//Specifies the page margins
section.PageSetup.Margins.All = 50f;

```

ASP.NET CORE

```

//Creates an instance of WordDocument Instance (Empty Word Document)
WordDocument document = new WordDocument();

```

```
//Adds a new section into the Word document
IWSection section = document.AddSection();
//Specifies the page margins
section.PageSetup.Margins.All = 50f;
```

XAMARIN

```
//Creates an instance of WordDocument Instance (Empty Word Document)
WordDocument document = new WordDocument();
//Adds a new section into the Word document
IWSection section = document.AddSection();
//Specifies the page margins
section.PageSetup.Margins.All = 50f;
```

All the textual contents in a Word document is represented by Paragraphs. Within the paragraph, textual contents are grouped into one or more child elements such as TextRange, field etc. Each TextRange represents a region of text with a common set of rich text formatting.

The following code example explains how to add a Paragraph into a Word document

C#

```
//Adds a new simple paragraph into the section
IWParagraph firstParagraph = section.AddParagraph();
//Sets the paragraph's horizontal alignment as justify
firstParagraph.ParagraphFormat.HorizontalAlignment =
HorizontalAlignment.Justify;
//Adds a text range into the paragraph
IWTextRange firstTextRange = firstParagraph.AppendText("AdventureWorks
Cycles,");
//sets the font formatting of the text range
firstTextRange.CharacterFormat.Bold = true;
firstTextRange.CharacterFormat.FontName = "Calibri";
firstTextRange.CharacterFormat.FontSize = 14;
//Adds another text range into the paragraph
IWTextRange secondTextRange = firstParagraph.AppendText(" the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.");
//sets the font formatting of the text range
secondTextRange.CharacterFormat.FontName = "Calibri";
secondTextRange.CharacterFormat.FontSize = 11;
```

VB.NET

```
'Adds a new simple paragraph into the section
Dim firstParagraph As IWParagraph = section.AddParagraph()
'Sets the paragraph's horizontal alignment as justify
firstParagraph.ParagraphFormat.HorizontalAlignment =
HorizontalAlignment.Justify
'Adds a text range into the paragraph
Dim firstTextRange As IWTextRange =
firstParagraph.AppendText("AdventureWorks Cycles, ")
'sets the font formatting of the text range
firstTextRange.CharacterFormat.Bold = True
firstTextRange.CharacterFormat.FontName = "Calibri"
```

```

firstTextRange.CharacterFormat.FontSize = 14
'Adds another text range into the paragraph
Dim secondTextRange As IWTextRange = firstParagraph.AppendText("the
fictitious company on which the AdventureWorks sample databases are based,
is a large, multinational manufacturing company.")
'sets the font formatting of the text range
secondTextRange.CharacterFormat.FontName = "Calibri"
secondTextRange.CharacterFormat.FontSize = 11

```

UWP

```

//Adds a new simple paragraph into the section
IWParagraph firstParagraph = section.AddParagraph();
//Sets the paragraph's horizontal alignment as justify
firstParagraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Justify;
//Adds a text range into the paragraph
IWTextRange firstTextRange = firstParagraph.AppendText("AdventureWorks
Cycles,");
//sets the font formatting of the text range
firstTextRange.CharacterFormat.Bold = true;
firstTextRange.CharacterFormat.FontName = "Calibri";
firstTextRange.CharacterFormat.FontSize = 14;
//Adds another text range into the paragraph
IWTextRange secondTextRange = firstParagraph.AppendText(" the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.");
//sets the font formatting of the text range
secondTextRange.CharacterFormat.FontName = "Calibri";
secondTextRange.CharacterFormat.FontSize = 11;

```

ASP.NET CORE

```

//Adds a new simple paragraph into the section
IWParagraph firstParagraph = section.AddParagraph();
//Sets the paragraph's horizontal alignment as justify
firstParagraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Justify;
//Adds a text range into the paragraph
IWTextRange firstTextRange = firstParagraph.AppendText("AdventureWorks
Cycles,");
//sets the font formatting of the text range
firstTextRange.CharacterFormat.Bold = true;
firstTextRange.CharacterFormat.FontName = "Calibri";
firstTextRange.CharacterFormat.FontSize = 14;
//Adds another text range into the paragraph
IWTextRange secondTextRange = firstParagraph.AppendText(" the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.");
//sets the font formatting of the text range
secondTextRange.CharacterFormat.FontName = "Calibri";
secondTextRange.CharacterFormat.FontSize = 11;

```

XAMARIN

```
//Adds a new simple paragraph into the section
IWParagraph firstParagraph = section.AddParagraph();
//Sets the paragraph's horizontal alignment as justify
firstParagraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Justify;
//Adds a text range into the paragraph
IWTextRange firstTextRange = firstParagraph.AppendText("AdventureWorks
Cycles,");
//sets the font formatting of the text range
firstTextRange.CharacterFormat.Bold = true;
firstTextRange.CharacterFormat.FontName = "Calibri";
firstTextRange.CharacterFormat.FontSize = 14;
//Adds another text range into the paragraph
IWTextRange secondTextRange = firstParagraph.AppendText(" the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.");
//sets the font formatting of the text range
secondTextRange.CharacterFormat.FontName = "Calibri";
secondTextRange.CharacterFormat.FontSize = 11;
```

The following code example shows how to add an image into the Word document.

C#

```
//Adds another paragraph and aligns it as center
IWParagraph paragraph = section.AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Center;
//Adds a picture into the paragraph
IWPicture picture =
paragraph.AppendPicture(Image.FromFile("DummyProfilePicture.jpg"));
//Specify the size of the picture
picture.Height = 100;
picture.Width = 100;
```

VB.NET

```
'Adds another paragraph and aligns it as center
Dim paragraph As IWParagraph = section.AddParagraph()
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Center
'Adds a picture into the paragraph
Dim picture As IWPicture =
paragraph.AppendPicture(Image.FromFile("DummyProfilePicture.jpg"))
'Specifies the size of the picture
picture.Height = 100
picture.Width = 100
```

UWP

```
//Adds another paragraph and aligns it as center
IWParagraph paragraph = section.AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
//Adds a picture into the paragraph
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
```



```
Stream imageStream1 =
assembly.GetManifestResourceStream("CreateWordSample.Assets.DummyProfilePicture.jpg");
IWPicture picture = paragraph.AppendPicture(imageStream1);
//Specify the size of the picture
picture.Height = 100;
picture.Width = 100;
```

ASP.NET CORE

```
//Adds another paragraph and aligns it as center
IWParagraph paragraph = section.AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
//Adds a picture into the paragraph
FileStream image1 = new FileStream("DummyProfilePicture.jpg", FileMode.Open,
FileAccess.Read);
IWPicture picture = paragraph.AppendPicture(image1);
//Specify the size of the picture
picture.Height = 100;
picture.Width = 100;
```

XAMARIN

```
//Adds another paragraph and aligns it as center
IWParagraph paragraph = section.AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
//Adds a picture into the paragraph
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream imageStream1 =
assembly.GetManifestResourceStream("CreateWordSample.Assets.DummyProfilePicture.jpg");
IWPicture picture = paragraph.AppendPicture(imageStream1);
//Specify the size of the picture
picture.Height = 100;
picture.Width = 100;
```

Table is another important element in Word that contains a set of paragraphs arranged in rows and columns. You can create simple as well as complex table by using Essential DocIO's API. The following code example creates a simple table and adds contents into it. Each table cell must contain at least one paragraph.

C#

```
//Adds a table into the Word document
IWTable table = section.AddTable();
//Creates the specified number of rows and columns
table.ResetCells(2, 2);
//Accesses the instance of the cell (first row, first cell)
WTableCell firstCell = table.Rows[0].Cells[0];
//Specifies the width of the cell
firstCell.Width = 150;
//Adds a paragraph into the cell; a cell must have atleast 1 paragraph
```

```

paragraph = firstCell.AddParagraph();
IWTextRange textRange = paragraph.AppendText("Profile picture");
textRange.CharacterFormat.Bold = true;
//Accesses the instance of cell (first row, second cell)
WTableCell secondCell = table.Rows[0].Cells[1];
secondCell.Width = 330;
paragraph = secondCell.AddParagraph();
textRange = paragraph.AppendText("Description");
textRange.CharacterFormat.Bold = true;
firstCell = table.Rows[1].Cells[0];
firstCell.Width = 150;
paragraph = firstCell.AddParagraph();
IWPicture profilePicture =
paragraph.AppendPicture(Image.FromFile(DummyProfile-Picture.jpg));
profilePicture.Height = 100;
profilePicture.Width = 100;
secondCell = table.Rows[1].Cells[1];
secondCell.Width = 330;
paragraph = secondCell.AddParagraph();
textRange = paragraph.AppendText("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.");

```

VB.NET

```

'Adds a table into the Word document
Dim table As ITable = section.AddTable()
'Creates the specified number of rows and columns
table.ResetCells(2, 2)
'Accesses the instance of the cell (first row, first cell)
Dim firstCell As WTableCell = table.Rows(0).Cells(0)
'Specifies the width of the cell
firstCell.Width = 150
'Adds a paragraph into the cell; a cell must have atleast 1 paragraph
paragraph = firstCell.AddParagraph()
Dim textRange As IWTextRange = paragraph.AppendText("Profile picture")
textRange.CharacterFormat.Bold = True
'Accesses the instance of cell (first row, second cell)
Dim secondCell As WTableCell = table.Rows(0).Cells(1)
secondCell.Width = 330
paragraph = secondCell.AddParagraph()
textRange = paragraph.AppendText("Description")
textRange.CharacterFormat.Bold = True
firstCell = table.Rows(1).Cells(0)
firstCell.Width = 150
paragraph = firstCell.AddParagraph()
Dim profilePicture As IWPicture =
paragraph.AppendPicture(Image.FromFile("DummyProfile-Picture.jpg"))
profilePicture.Height = 100
profilePicture.Width = 100
secondCell = table.Rows(1).Cells(1)
secondCell.Width = 330
paragraph = secondCell.AddParagraph()
textRange = paragraph.AppendText("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.")

```

UWP

```
//Adds a table into the Word document
IWTable table = section.AddTable();
//Creates the specified number of rows and columns
table.ResetCells(2, 2);
//Accesses the instance of the cell (first row, first cell)
WTableCell firstCell = table.Rows[0].Cells[0];
//Specifies the width of the cell
firstCell.Width = 150;
//Adds a paragraph into the cell; a cell must have atleast 1 paragraph
paragraph = firstCell.AddParagraph();
IWTextRange textRange = paragraph.AppendText("Profile picture");
textRange.CharacterFormat.Bold = true;
//Accesses the instance of cell (first row, second cell)
WTableCell secondCell = table.Rows[0].Cells[1];
secondCell.Width = 330;
paragraph = secondCell.AddParagraph();
textRange = paragraph.AppendText("Description");
textRange.CharacterFormat.Bold = true;
firstCell = table.Rows[1].Cells[0];
firstCell.Width = 150;
paragraph = firstCell.AddParagraph();
Stream imageStream2 =
assembly.GetManifestResourceStream("CreateWordSample.Assets.DummyProfile-
Picture.jpg");
IWPicture profilePicture = paragraph.AppendPicture(imageStream2);
profilePicture.Height = 100;
profilePicture.Width = 100;
secondCell = table.Rows[1].Cells[1];
secondCell.Width = 330;
paragraph = secondCell.AddParagraph();
textRange = paragraph.AppendText("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.");
```

ASP.NET CORE

```
//Adds a table into the Word document
IWTable table = section.AddTable();
//Creates the specified number of rows and columns
table.ResetCells(2, 2);
//Accesses the instance of the cell (first row, first cell)
WTableCell firstCell = table.Rows[0].Cells[0];
//Specifies the width of the cell
firstCell.Width = 150;
//Adds a paragraph into the cell; a cell must have atleast 1 paragraph
paragraph = firstCell.AddParagraph();
IWTextRange textRange = paragraph.AppendText("Profile picture");
textRange.CharacterFormat.Bold = true;
//Accesses the instance of cell (first row, second cell)
WTableCell secondCell = table.Rows[0].Cells[1];
secondCell.Width = 330;
paragraph = secondCell.AddParagraph();
textRange = paragraph.AppendText("Description");
```

```

textRange.CharacterFormat.Bold = true;
firstCell = table.Rows[1].Cells[0];
firstCell.Width = 150;
paragraph = firstCell.AddParagraph();
FileStream image2 = new FileStream("DummyProfile-Picture.jpg",
    FileMode.Open, FileAccess.Read);
IWPicture profilePicture = paragraph.AppendPicture(image2);
profilePicture.Height = 100;
profilePicture.Width = 100;
secondCell = table.Rows[1].Cells[1];
secondCell.Width = 330;
paragraph = secondCell.AddParagraph();
textRange = paragraph.AppendText("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.");

```

XAMARIN

```

//Adds a table into the Word document
IWTable table = section.AddTable();
//Creates the specified number of rows and columns
table.ResetCells(2, 2);
//Accesses the instance of the cell (first row, first cell)
WTableCell firstCell = table.Rows[0].Cells[0];
//Specifies the width of the cell
firstCell.Width = 150;
//Adds a paragraph into the cell; a cell must have atleast 1 paragraph
paragraph = firstCell.AddParagraph();
IWTextRange textRange = paragraph.AppendText("Profile picture");
textRange.CharacterFormat.Bold = true;
//Accesses the instance of cell (first row, second cell)
WTableCell secondCell = table.Rows[0].Cells[1];
secondCell.Width = 330;
paragraph = secondCell.AddParagraph();
textRange = paragraph.AppendText("Description");
textRange.CharacterFormat.Bold = true;
firstCell = table.Rows[1].Cells[0];
firstCell.Width = 150;
paragraph = firstCell.AddParagraph();
Stream imageStream2 =
assembly.GetManifestResourceStream("CreateWordSample.Assets.DummyProfile-
Picture.jpg");
IWPicture profilePicture = paragraph.AppendPicture(imageStream2);
profilePicture.Height = 100;
profilePicture.Width = 100;
secondCell = table.Rows[1].Cells[1];
secondCell.Width = 330;
paragraph = secondCell.AddParagraph();
textRange = paragraph.AppendText("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.");

```

Essential DocIO allow you to create simple and multi-level lists. The following code snippet explains about how to create a numbered and bulleted list.

C#

```

//Writes default numbered list.
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
//Applies the default numbered list formats
paragraph.ListFormat.ApplyDefNumberedStyle();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 1");
//Specifies the list format to continue from last list
paragraph.ListFormat.ContinueListNumbering();
//Increments the list level
paragraph.ListFormat.IncreaseIndentLevel();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
paragraph.ListFormat.ContinueListNumbering();
//Decrements the list level
paragraph.ListFormat.DecreaseIndentLevel();
section.AddParagraph();
//Writes default bulleted list.
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
//Applies the default bulleted list formats
paragraph.ListFormat.ApplyDefBulletStyle();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 1");
//Specifies the list format to continue from last list
paragraph.ListFormat.ContinueListNumbering();
//Increments the list level
paragraph.ListFormat.IncreaseIndentLevel();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
//Specifies the list format to continue from last list
paragraph.ListFormat.ContinueListNumbering();
//Decrements the list level
paragraph.ListFormat.DecreaseIndentLevel();
section.AddParagraph();

```

VB.NET

```

'Writes default numbered list.
paragraph = section.AddParagraph()
paragraph.AppendText("Level 0")
'Applies the default numbered list formats
paragraph.ListFormat.ApplyDefNumberedStyle()
paragraph = section.AddParagraph()
paragraph.AppendText("Level 1")
'Specifies the list format to continue from last list
paragraph.ListFormat.ContinueListNumbering()
'Increments the list level
paragraph.ListFormat.IncreaseIndentLevel()
paragraph = section.AddParagraph()
paragraph.AppendText("Level 0")
paragraph.ListFormat.ContinueListNumbering()
'Decrements the list level
paragraph.ListFormat.DecreaseIndentLevel()

```

```

section.AddParagraph()
'Writes default bulleted list.'
paragraph = section.AddParagraph()
paragraph.AppendText("Level 0")
'Applies the default bulleted list formats'
paragraph.ListFormat.ApplyDefBulletStyle()
paragraph = section.AddParagraph()
paragraph.AppendText("Level 1")
'Specifies the list format to continue from last list'
paragraph.ListFormat.ContinueListNumbering()
'Increments the list level'
paragraph.ListFormat.IncreaseIndentLevel()
paragraph = section.AddParagraph()
paragraph.AppendText("Level 0")
'Specifies the list format to continue from last list'
paragraph.ListFormat.ContinueListNumbering()
'Decrements the list level'
paragraph.ListFormat.DecreaseIndentLevel()
section.AddParagraph()

```

UWP

```

//Writes default numbered list.
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
//Applies the default numbered list formats
paragraph.ListFormat.ApplyDefNumberedStyle();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 1");
//Specifies the list format to continue from last list
paragraph.ListFormat.ContinueListNumbering();
//Increments the list level
paragraph.ListFormat.IncreaseIndentLevel();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
paragraph.ListFormat.ContinueListNumbering();
//Decrements the list level
paragraph.ListFormat.DecreaseIndentLevel();
section.AddParagraph();
//Writes default bulleted list.
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
//Applies the default bulleted list formats
paragraph.ListFormat.ApplyDefBulletStyle();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 1");
//Specifies the list format to continue from last list
paragraph.ListFormat.ContinueListNumbering();
//Increments the list level
paragraph.ListFormat.IncreaseIndentLevel();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
//Specifies the list format to continue from last list
paragraph.ListFormat.ContinueListNumbering();
//Decrements the list level
paragraph.ListFormat.DecreaseIndentLevel();

```

```
section.AddParagraph();
```

ASP.NET CORE

```
//Writes default numbered list.
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
//Applies the default numbered list formats
paragraph.ListFormat.ApplyDefNumberedStyle();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 1");
//Specifies the list format to continue from last list
paragraph.ListFormat.ContinueListNumbering();
//Increments the list level
paragraph.ListFormat.IncreaseIndentLevel();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
paragraph.ListFormat.ContinueListNumbering();
//Decrements the list level
paragraph.ListFormat.DecreaseIndentLevel();
section.AddParagraph();
//Writes default bulleted list.
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
//Applies the default bulleted list formats
paragraph.ListFormat.ApplyDefBulletStyle();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 1");
//Specifies the list format to continue from last list
paragraph.ListFormat.ContinueListNumbering();
//Increments the list level
paragraph.ListFormat.IncreaseIndentLevel();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
//Specifies the list format to continue from last list
paragraph.ListFormat.ContinueListNumbering();
//Decrements the list level
paragraph.ListFormat.DecreaseIndentLevel();
section.AddParagraph();
```

XAMARIN

```
//Writes default numbered list.
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
//Applies the default numbered list formats
paragraph.ListFormat.ApplyDefNumberedStyle();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 1");
//Specifies the list format to continue from last list
paragraph.ListFormat.ContinueListNumbering();
//Increments the list level
paragraph.ListFormat.IncreaseIndentLevel();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
paragraph.ListFormat.ContinueListNumbering();
```

```
//Decrements the list level
paragraph.ListFormat.DecreaseIndentLevel();
section.AddParagraph();
//Writes default bulleted list.
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
//Applies the default bulleted list formats
paragraph.ListFormat.ApplyDefBulletStyle();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 1");
//Specifies the list format to continue from last list
paragraph.ListFormat.ContinueListNumbering();
//Increments the list level
paragraph.ListFormat.IncreaseIndentLevel();
paragraph = section.AddParagraph();
paragraph.AppendText("Level 0");
//Specifies the list format to continue from last list
paragraph.ListFormat.ContinueListNumbering();
//Decrements the list level
paragraph.ListFormat.DecreaseIndentLevel();
section.AddParagraph();
```

Finally, save the document in file system and close its instance.

C#

```
//Saves the document in the given name and format
document.Save(outputFileName, FormatType.Docx);
//Releases the resources occupied by WordDocument instance
document.Close();
```

VB.NET

```
'Saves the document in the given name and format
document.Save(outputFileName, FormatType.Docx)
'Releases the resources occupied by WordDocument instance
document.Close()
```

UWP

```
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word document file in local machine
Save(stream, outputFileName);
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
```

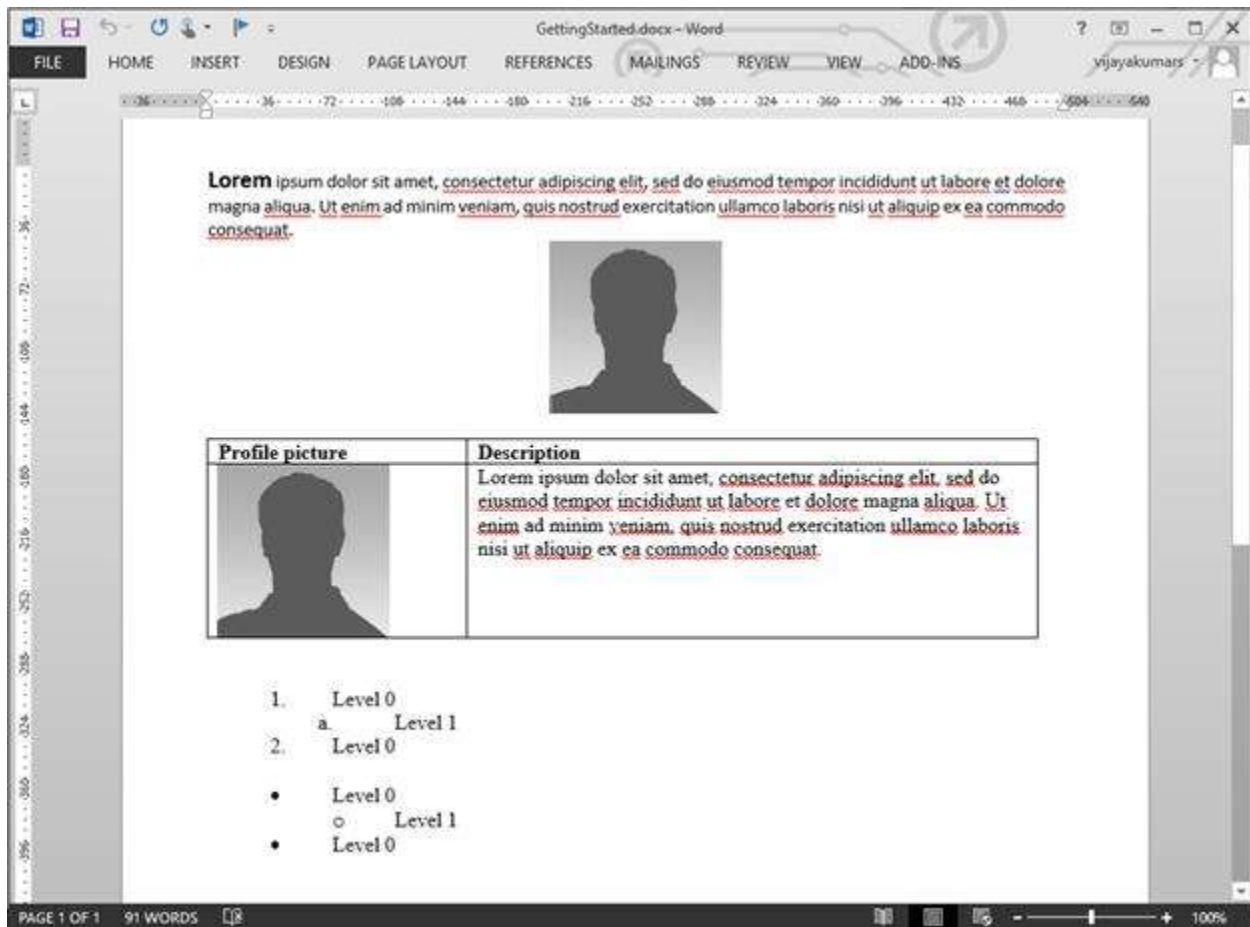


```
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", outputFileName);
```

XAMARIN

```
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(outputFileName,
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

The resultant Word document looks as follows.



Modifying an existing Word document

Essential DocIO allows you to manipulate an existing Word document, RTF, WordML, HTML and Plain text files. You can modify the documents either by manipulating DocIO's DOM or by using DocIO's built-in functionalities such as Find and Replace, replacing bookmark contents etc.

Here, you can see how an existing Word document is loaded into DocIO's DOM, replaces an existing content with another and finally saves the Word document.

You can open an existing Word document either by using constructor of `WordDocument` class or by using `Open` method of `WordDocument` class that reads the document and populates DocIO's DOM. The following code example shows how to load an existing document.

C#

```
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument("Giant Panda.docx");
//Replaces the word "bear" as "panda"
document.Replace("bear", "panda", false, true);
//Saves the Word document
document.Save("Result.docx");
//Closes the document
document.Close();
```

VB.NET

```
'Loads an existing Word document into DocIO instance
Dim document As New WordDocument("Giant Panda.docx")
'Replaces the word "bear" as "panda"
document.Replace("bear", "panda", False, True)
'Saves the Word document
document.Save("Result.docx")
'Closes the document
document.Close()
```

UWP

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream FileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Giant
Panda.docx");
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(FileStream);
//Replaces the word "bear" as "panda"
document.Replace("bear", "panda", false, true);
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word document file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```

FileStream fileStream = new FileStream(@"Giant
Panda.docx", FileMode.Open, FileAccess.ReadWrite);
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(fileStream, FormatType.Automatic);
//Replaces the word "bear" as "panda"
document.Replace("bear", "panda", false, true);
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
stream.Position = 0;
document.Close();
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream FileStream =
assembly.GetManifestResourceStream("XamarinFormsAppl.Assets.Giant
Panda.docx");
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(FileStream, FormatType.Automatic);
//Replaces the word "bear" as "panda"
document.Replace("bear", "panda", false, true);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example explains how to search a particular text and highlight it.

C#

```

//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(@"..\..\Data\Giant Panda.docx");
//Finds the occurrence of the Word "panda" in the document
TextSelection[] textSelection = document.FindAll("panda", false, true);
//Iterates through each occurrence and highlights it
foreach (TextSelection selection in textSelection)
{
    IWTextRange textRange = selection.GetAsOneRange();
    textRange.CharacterFormat.HighlightColor = Color.Yellow;
}
//Saves and closes the document
document.Save("Result.docx");
document.Close();

```

VB.NET

```

'Loads an existing Word document into DocIO instance
Dim document As New WordDocument("../Data/Giant Panda.docx")
'Finds the occurrence of the word "panda" in the document
Dim textSelection As TextSelection() = document.FindAll("panda", False,
True)
'Iterates through each occurrence and highlights it.
For Each selection As TextSelection In textSelection
Dim textRange As ITextRange = selection.GetAsOneRange()
textRange.CharacterFormat.HighlightColor = Color.Yellow
Next
document.Save("Result.docx")
document.Close()

```

UWP

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream FileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Test.docx");
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(FileStream);
//Finds the occurrence of the Word "panda" in the document
TextSelection[] textSelection = document.FindAll("panda", false, true);
//Iterates through each occurrence and highlights it
foreach (TextSelection selection in textSelection)
{
ITextRange textRange = selection.GetAsOneRange();
textRange.CharacterFormat.HighlightColor = Color.Yellow;
}
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

FileStream fileStream = new
FileStream(@"Test.docx", FileMode.Open, FileAccess.ReadWrite);
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(fileStream, FormatTypeAutomatic);
//Finds the occurrence of the Word "panda" in the document
TextSelection[] textSelection = document.FindAll("panda", false, true);
//Iterates through each occurrence and highlights it
foreach (TextSelection selection in textSelection)
{
ITextRange textRange = selection.GetAsOneRange();
textRange.CharacterFormat.HighlightColor = Syncfusion.Drawing.Color.Yellow;
}
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream

```

```
document.Save(stream, FormatType.Docx);
stream.Position = 0;
document.Close();
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream FileStream =
assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Test.docx");
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(FileStream, FormatType.Automatic);
//Finds the occurrence of the Word "panda" in the document
TextSelection[] textSelection = document.FindAll("panda", false, true);
//Iterates through each occurrence and highlights it
foreach (TextSelection selection in textSelection)
{
    IWTextRange textRange = selection.GetAsOneRange();
    textRange.CharacterFormat.HighlightColor = Syncfusion.Drawing.Color.Yellow;
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("GettingStartedSample.docx", "application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
```

Performing Mail merge

Essential DocIO allows to generate documents by filling data in template document from data source. Mail merge operation automatically maps the column name in the data source and names of the merge fields in the template Word document and fills the data.

The following data sources are supported by Essential DocIO for performing Mail merge.

- String Arrays
- ADO.NET objects
- Business Objects
- Dynamic objects

Also, you can perform more than one Mail merge operations over the same template to generate document as per your requirement.

Follow the given steps to perform simple Mail merge in a Word document.

Let's consider that you have a template Word document with merge fields as shown.



The **MailMerge** class provides various overloads for **Execute** method to perform Mail merge from various data source. The Mail merge operation replaces the matching merge fields with the respective data.

The following code example shows how to perform simple Mail merge by using string array.

C#

```
//Loads the template document with required merge fields
WordDocument document = new
WordDocument(@"..\..\data\SimpleMailMergeTemplate.docx");
//Initializes the string array with field names
string[] fieldNames = new string[] { "FirstName", "LastName", "Email",
"Country" };
//Initializes the string array with field values
string[] fieldValues = new string[] { "John", "Smith",
"john_smith@domain.com", "USA" };
//Executes the mail merge operation that replaces the matching field names
with field values respectively.
document.MailMerge.Execute(fieldNames, fieldValues);
//Saves and closes the WordDocument instance
document.Save("result.docx");
document.Close();
```

VB.NET

```
'Loads the template document with required merge fields
Dim document As New WordDocument(@"..\..\data\SimpleMailMergeTemplate.docx")
'Initializes the string array with field names
Dim fieldNames As String() = New String() { "FirstName", "LastName", "Email",
"Country" }
'Initializes the string array with field values
Dim fieldValues As String() = New String() { "John", "Smith",
"john_smith@domain.com", "USA" }
'Executes the mail merge operation that replaces the matching field names
with field values respectively.
document.MailMerge.Execute(fieldNames, fieldValues)
'Saves and closes the WordDocument instance
document.Save("result.docx")
document.Close()
```

UWP

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream FileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.SimpleMailMergeT
emplate.docx");
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(FileStream);
//Initializes the string array with field names
string[] fieldNames = new string[] { "FirstName", "LastName", "Email",
"Country" };
//Initializes the string array with field values
string[] fieldValues = new string[] { "John", "Smith",
"john_smith@domain.com", "USA" };
//Executes the mail merge operation that replaces the matching field names
with field values respectively.
document.MailMerge.Execute(fieldNames, fieldValues);
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word document file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

FileStream fileStream = new FileStream(@"SimpleMailMergeTemplate.docx",
FileMode.Open, FileAccess.ReadWrite);
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(fileStream, FormatType.Automatic);
//Initializes the string array with field names
string[] fieldNames = new string[] { "FirstName", "LastName", "Email",
"Country" };
//Initializes the string array with field values
string[] fieldValues = new string[] { "John", "Smith",
"john_smith@domain.com", "USA" };
//Executes the mail merge operation that replaces the matching field names
with field values respectively.
document.MailMerge.Execute(fieldNames, fieldValues);
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
stream.Position = 0;
document.Close();
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

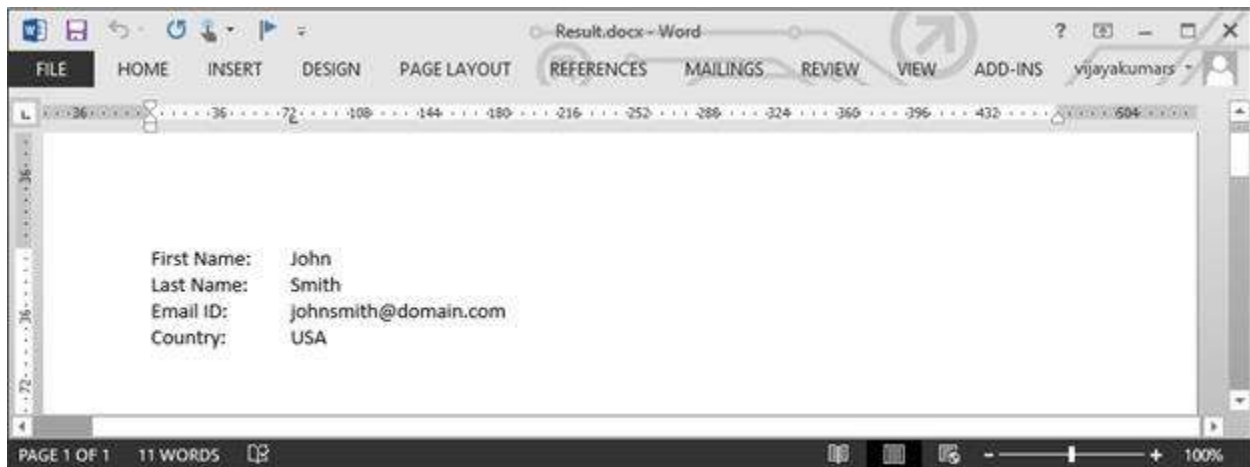
```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream FileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.SimpleMailMergeT
emplate.docx");
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(FileStream, FormatType.Automatic);

```

```
//Initializes the string array with field names
string[] fieldNames = new string[] { "FirstName", "LastName", "Email",
"Country" };
//Initializes the string array with field values
string[] fieldValues = new string[] { "John", "Smith",
"john_smith@domain.com", "USA" };
//Executes the mail merge operation that replaces the matching field names
with field values respectively.
document.MailMerge.Execute(fieldNames, fieldValues);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("GettingStartedSample.docx", "application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
```

The resultant Word document look as follows.



Simple Mail merge with Group

You can perform Mail merge with group to append multiple records from data source into a single document. Group is a part of the document enclosed by two special merge fields named «TableStart:TableName» and «TableEnd:TableName»

- «TableStart:TableName» - denotes the start of the group
- «TableEnd:TableName» - denotes the end of the group

The region between these two merge fields get repeated for every record from the data source.

For example – let's consider that you have a template document as shown.



Here, in this template, Employees is the group name and exact same name should be used while performing Mail merge through code. There are two special merge fields “TableStart:Employees” and “TableEnd:Employees”, to denote the start and end of the Mail merge group.

To merge an image in the replace of a merge field, you need to add a prefix (“Image:”)the merge field name.

For example: the merge field name should be like “<<Image:Photo>>”(<<Image:MergeFieldName>>)

The following code example shows how to perform Mail merge with objects.

C#

```
//Loads the template document
WordDocument document = new
WordDocument(@"..\..\Data\EmployeesTemplate.doc");
//Gets the employee details as IEnumerable collection
List<Employee> employeeList = GetEmployees();
//Creates an instance of MailMergeDataTable by specifying MailMerge group
name and IEnumerable collection
MailMergeDataTable dataSource = new MailMergeDataTable("Employees",
employeeList);
//Performs Mail merge
document.MailMerge.ExecuteGroup(dataSource);
//Saves and closes the document
document.Save("Result.docx");
document.Close();
```

VB.NET

```
//Loads the template document
Dim document As New WordDocument(@"..\..\Data\EmployeesTemplate.doc")
'Gets the employee details as IEnumerable collection
Dim employeeList As List(Of Employee) = GetEmployees()
'Creates an instance of MailMergeDataTable by specifying MailMerge group
name and IEnumerable collection
Dim dataSource As New MailMergeDataTable("Employees", employeeList)
```

```
'Performs Mail merge
document.MailMerge.ExecuteGroup(dataSource)
'Saves and closes the document
document.Save("Result.docx")
document.Close()
```

UWP

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream FileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.EmployeesTemplate.docx");
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(FileStream);
//Gets the employee details as IEnumerable collection
List<Employee> employeeList = GetEmployees();
//Creates an instance of MailMergeDataTable by specifying MailMerge group name and IEnumerable collection
MailMergeDataTable dataSource = new MailMergeDataTable("Employees", employeeList);
//Performs Mail merge
document.MailMerge.ExecuteGroup(dataSource);
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word document file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
FileStream fileStream = new FileStream(@"EmployeesTemplate.docx",
 FileMode.Open, FileAccess.ReadWrite);
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(fileStream, FormatType.Automatic);
//Gets the employee details as IEnumerable collection
List<Employee> employeeList = GetEmployees();
//Creates an instance of MailMergeDataTable by specifying MailMerge group name and IEnumerable collection
MailMergeDataTable dataSource = new MailMergeDataTable("Employees", employeeList);
//Performs Mail merge
document.MailMerge.ExecuteGroup(dataSource);
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream FileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.EmployeesTemplate.docx");
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(FileStream, FormatType.Automatic);
//Gets the employee details as IEnumerable collection
List<Employee> employeeList = GetEmployees();
//Creates an instance of MailMergeDataTable by specifying MailMerge group name and IEnumerable collection
MailMergeDataTable dataSource = new MailMergeDataTable("Employees", employeeList);
//Performs Mail merge
document.MailMerge.ExecuteGroup(dataSource);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx", "application/msword", stream);
//Please download the helper files from the below link to save the stream as file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin

```

The following code example provides supporting methods and class for the above code

C#

```

public List<Employee> GetEmployees()
{
List<Employee> employees = new List<Employee>();
employees.Add(new Employee("Nancy", "Smith", "Sales Representative", "505 - 20th Ave. E. Apt. 2A,", "Seattle", "WA", "USA", "Nancy.png"));
employees.Add(new Employee("Andrew", "Fuller", "Vice President, Sales", "908 W. Capital Way", "Tacoma", "WA", "USA", "Andrew.png"));
employees.Add(new Employee("Roland", "Mendel", "Sales Representative", "722 Moss Bay Blvd.", "Kirkland", "WA", "USA", "Janet.png"));
employees.Add(new Employee("Margaret", "Peacock", "Sales Representative", "4110 Old Redmond Rd.", "Redmond", "WA", "USA", "Margaret.png"));
employees.Add(new Employee("Steven", "Buchanan", "Sales Manager", "14 Garrett Hill", "London", string.Empty, "UK", "Steven.png"));
return employees;
}

public class Employee
{
public string FirstName { get; set; }
public string LastName { get; set; }
public string Address { get; set; }
public string City { get; set; }
public string Region { get; set; }
public string Country { get; set; }
public string Title { get; set; }
}

```

```

public Image Photo { get; set; }
public Employee(string firstName, string lastName, string title, string
address, string city, string region, string country, string photoFilePath)
{
    FirstName = firstName;
    LastName = lastName;
    Title = title;
    Address = address;
    City = city;
    Region = region;
    Country = country;
    Photo = Image.FromFile(photoFilePath);
}
}

```

VB.NET

```

Public Function GetEmployees() As List(Of Employee)
    Dim employees As New List(Of Employee) ()
    employees.Add(New Employee("Nancy", "Smith", "Sales Representative", "505 -
20th Ave. E. Apt. 2A", "Seattle", "WA", "USA", "Nancy.png"))
    employees.Add(New Employee("Andrew", "Fuller", "Vice President, Sales", "908
W. Capital Way", "Tacoma", "WA", "USA", "Andrew.png"))
    employees.Add(New Employee("Janet", "Leverling", "Sales Representative",
"722 Moss Bay Blvd.", "Kirkland", "WA", "USA", "Janet.png"))
    employees.Add(New Employee("Margaret", "Peacock", "Sales Representative",
"4110 Old Redmond Rd.", "Redmond", "WA", "USA", "Margaret.png"))
    employees.Add(New Employee("Steven", "Buchanan", "Sales Manager", "14
Garrett Hill", "London", String.Empty, "UK", "Steven.png"))
    Return employees
End Function

Public Class Employee
    Public Property FirstName() As String
    Get
        Return m_FirstName
    End Get
    Set(value As String)
        m_FirstName = value
    End Set
    End Property
    Private m_FirstName As String
    Public Property LastName() As String
    Get
        Return m_LastName
    End Get
    Set(value As String)
        m_LastName = value
    End Set
    End Property
    Private m_LastName As String
    Public Property Address() As String
    Get
        Return m_Address
    End Get
    Set(value As String)
        m_Address = value
    End Set

```

```
End Set
End Property
Private m_Address As String
Public Property City() As String
Get
Return m_City
End Get
Set(value As String)
m_City = value
End Set
End Property
Private m_City As String
Public Property Region() As String
Get
Return m_Region
End Get
Set(value As String)
m_Region = value
End Set
End Property
Private m_Region As String
Public Property Country() As String
Get
Return m_Country
End Get
Set(value As String)
m_Country = value
End Set
End Property
Private m_Country As String
Public Property Title() As String
Get
Return m_Title
End Get
Set(value As String)
m_Title = value
End Set
End Property
Private m_Title As String
Public Property Photo() As Image
Get
Return m_Photo
End Get
Set(value As Image)
m_Photo = value
End Set
End Property
Private m_Photo As Image
Public Sub New(firstName As String, lastName As String, title As String,
address As String, city As String, region As String, country As String,
photoFilePath As String)
firstName = firstName
lastName = lastName
title = title
address = address
city = city
region = region
```

```
country = country
Photo = Image.FromFile(photoFilePath)
End Sub
End Class
```

UWP

```
public List<Employee> GetEmployees()
{
    List<Employee> employees = new List<Employee>();
    employees.Add(new Employee("Nancy", "Smith", "Sales Representative", "505 - 20th Ave. E. Apt. 2A,", "Seattle", "WA", "USA", "Nancy.png"));
    employees.Add(new Employee("Andrew", "Fuller", "Vice President, Sales", "908 W. Capital Way", "Tacoma", "WA", "USA", "Andrew.png"));
    employees.Add(new Employee("Roland", "Mendel", "Sales Representative", "722 Moss Bay Blvd.", "Kirkland", "WA", "USA", "Janet.png"));
    employees.Add(new Employee("Margaret", "Peacock", "Sales Representative", "4110 Old Redmond Rd.", "Redmond", "WA", "USA", "Margaret.png"));
    employees.Add(new Employee("Steven", "Buchanan", "Sales Manager", "14 Garrett Hill", "London", string.Empty, "UK", "Steven.png"));
    return employees;
}

public class Employee
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Address { get; set; }
    public string City { get; set; }
    public string Region { get; set; }
    public string Country { get; set; }
    public string Title { get; set; }
    public Syncfusion.Drawing.Image Photo { get; set; }
    public Employee(string firstName, string lastName, string title, string address, string city, string region, string country, string photoFilePath)
    {
        FirstName = firstName;
        LastName = lastName;
        Title = title;
        Address = address;
        City = city;
        Region = region;
        Country = country;
        Stream stream =
        typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("CreateWordSample.Assets." + photoFilePath);
        Photo = Syncfusion.Drawing.Image.FromStream(stream);
    }
}
```

ASP.NET CORE

```
public List<Employee> GetEmployees()
{
    List<Employee> employees = new List<Employee>();
    employees.Add(new Employee("Nancy", "Smith", "Sales Representative", "505 - 20th Ave. E. Apt. 2A,", "Seattle", "WA", "USA", "Nancy.png"));
}
```

```

employees.Add(new Employee("Andrew", "Fuller", "Vice President, Sales", "908
W. Capital Way", "Tacoma", "WA", "USA", "Andrew.png"));
employees.Add(new Employee("Roland", "Mendel", "Sales Representative", "722
Moss Bay Blvd.", "Kirkland", "WA", "USA", "Janet.png"));
employees.Add(new Employee("Margaret", "Peacock", "Sales Representative",
"4110 Old Redmond Rd.", "Redmond", "WA", "USA", "Margaret.png"));
employees.Add(new Employee("Steven", "Buchanan", "Sales Manager", "14
Garrett Hill", "London", string.Empty, "UK", "Steven.png"));
return employees;
}
public class Employee
{
public string FirstName { get; set; }
public string LastName { get; set; }
public string Address { get; set; }
public string City { get; set; }
public string Region { get; set; }
public string Country { get; set; }
public string Title { get; set; }
public Syncfusion.Drawing.Image Photo { get; set; }
public Employee(string firstName, string lastName, string title, string
address, string city, string region, string country, string photoFilePath)
{
    FirstName = firstName;
    LastName = lastName;
    Title = title;
    Address = address;
    City = city;
    Region = region;
    Country = country;
    FileStream imageStream = new FileStream(photoFilePath, FileMode.Open,
    FileAccess.ReadWrite);
    Photo = Syncfusion.Drawing.Image.FromStream(imageStream);
    imageStream.Dispose();
    imageStream.Close();
}
}

```

XAMARIN

```

public List<Employee> GetEmployees()
{
    List<Employee> employees = new List<Employee>();
    employees.Add(new Employee("Nancy", "Smith", "Sales Representative", "505 -
20th Ave. E. Apt. 2A,", "Seattle", "WA", "USA", "Nancy.png"));
    employees.Add(new Employee("Andrew", "Fuller", "Vice President, Sales", "908
W. Capital Way", "Tacoma", "WA", "USA", "Andrew.png"));
    employees.Add(new Employee("Roland", "Mendel", "Sales Representative", "722
Moss Bay Blvd.", "Kirkland", "WA", "USA", "Janet.png"));
    employees.Add(new Employee("Margaret", "Peacock", "Sales Representative",
"4110 Old Redmond Rd.", "Redmond", "WA", "USA", "Margaret.png"));
    employees.Add(new Employee("Steven", "Buchanan", "Sales Manager", "14
Garrett Hill", "London", string.Empty, "UK", "Steven.png"));
    return employees;
}
public class Employee

```

```
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Address { get; set; }
    public string City { get; set; }
    public string Region { get; set; }
    public string Country { get; set; }
    public string Title { get; set; }
    public Syncfusion.Drawing.Image Photo { get; set; }
    public Employee(string firstName, string lastName, string title, string
address, string city, string region, string country, string photoFilePath)
    {
        FirstName = firstName;
        LastName = lastName;
        Title = title;
        Address = address;
        City = city;
        Region = region;
        Country = country;
        Stream stream =
        typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("CreateWordSamp
le.Assets." + photoFilePath);
        Photo = Syncfusion.Drawing.Image.FromStream(stream);
    }
}
```

The resultant document looks as follows.



Converting Word document to PDF

Essential DocIO allows you to convert a Word document into PDF document in a few lines of code.

For converting a Word document to PDF, the following assemblies are required to be referenced in your application

- Syncfusion.DocIO.Base
- Syncfusion.OfficeChart.Base
- Syncfusion.Compression.Base
- Syncfusion.Pdf.Base
- Syncfusion.DocToPdfConverter.Base
- Syncfusion.OfficeChartToImageConverter.WPF
- Syncfusion.SfChart.WPF

For converting a word document to PDF in Xamarin, UWP and ASP.NET Core platform, the following assemblies are required.

- Syncfusion.DocIO.Portable
- Syncfusion.OfficeChart.Portable
- Syncfusion.Compression.Portable

- Syncfusion.Pdf.Portable
- Syncfusion.DocIORenderer.Portable

DocToPDFConverter class is responsible for converting a Word document into PDF.

In portable projects, **DocIORenderer** is responsible for converting a Word document into PDF.

The following code example illustrates how to convert a Word document into PDF document.

C#

```
//Loads the template document
WordDocument wordDocument = new WordDocument(inputWordDocument,
FormatTypeAutomatic );
//Initializes chart to image converter for converting charts during Word to
pdf conversion
wordDocument.ChartToImageConverter = new ChartToImageConverter();
wordDocument.ChartToImageConverter.ScalingMode = ScalingMode.Normal;
//Creates an instance of DocToPDFConverter - responsible for Word to PDF
conversion
DocToPDFConverter converter = new DocToPDFConverter();
//Converts Word document into PDF document
PdfDocument pdfDocument = converter.ConvertToPDF(wordDocument);
//Saves the PDF file to file system
pdfDocument.Save("Sample.pdf");
//closes the instance of document objects
pdfDocument.Close();
wordDocument.Close();
```

VB.NET

```
'Loads the template document
Dim wordDocument As New WordDocument(inputWordDocument,
FormatTypeAutomatic)
'Initializes chart to image converter for converting charts during Word to
pdf conversion
wordDocument.ChartToImageConverter = New ChartToImageConverter()
wordDocument.ChartToImageConverter.ScalingMode = ScalingMode.Normal
'Creates an instance of DocToPDFConverter - responsible for Word to PDF
conversion
Dim converter As New DocToPDFConverter()
'Converts Word document into PDF document
Dim pdfDocument As PdfDocument = converter.ConvertToPDF(wordDocument)
'Saves the PDF file to file system
pdfDocument.Save("Sample.pdf")
'closes the instance of document objects
pdfDocument.Close()
wordDocument.Close()
```

UWP

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputWordDocument =
assembly.GetManifestResourceStream("CreateWordSample.Assets.WordToPDF.docx");
;
//Loads the template document
```

```

WordDocument wordDocument = new WordDocument(inputWordDocument,
FormatType.Automatic);
//Creates an instance of DocToPDFConverter - responsible for Word to PDF
conversion
DocIORenderer converter = new DocIORenderer();
//Converts Word document into PDF document
PdfDocument pdfDocument = converter.ConvertToPDF(wordDocument);
//Save the document into stream.
MemoryStream outputStream = new MemoryStream();
pdfDocument.Save(outputStream);
//Closes the instance of PDF document object
pdfDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(outputStream, "Output.pdf");
//Saves the Word document
async void Save(MemoryStream streams, string filename)
{
streams.Position = 0;
StorageFile stFile;
if
(! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
{
FileSavePicker savePicker = new FileSavePicker();
savePicker.DefaultFileExtension = ".pdf";
savePicker.SuggestedFileName = filename;
savePicker.FileTypeChoices.Add("PDF Documents", new List<string>() { ".pdf"
});
stFile = await savePicker.PickSaveFileAsync();
}
else
{
StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
stFile = await local.CreateFileAsync(filename,
CreationCollisionOption.ReplaceExisting);
}
if (stFile != null)
{
using (IRandomAccessStream zipStream = await
stFile.OpenAsync(FileAccessMode.ReadWrite))
{
//Write compressed data from memory to file
using (Stream outstream = zipStream.AsStreamForWrite())
{
byte[] buffer = streams.ToArray();
outstream.Write(buffer, 0, buffer.Length);
outstream.Flush();
}
}
}
//Launch the saved Word file
await Windows.System.Launcher.LaunchFileAsync(stFile);
}

```

ASP.NET CORE

```

FileStream fileStream = new FileStream(@"EmployeesTemplate.docx",
    FileMode.Open, FileAccess.ReadWrite);
//Loads an existing Word document into DocIO instance
WordDocument wordDocument = new WordDocument(fileStream,
    FormatType.Automatic);
//Creates an instance of DocToPDFConverter - responsible for Word to PDF
conversion
DocIORenderer converter = new DocIORenderer();
//Converts Word document into PDF document
PdfDocument pdfDocument = converter.ConvertToPDF(wordDocument);
//Save the document into stream.
MemoryStream outputStream = new MemoryStream();
pdfDocument.Save(outputStream);
//Closes the instance of PDF document object
pdfDocument.Close();
wordDocument.Close();
outputStream.Position = 0;
//Download Word document in the browser
return File(outputStream, "application/pdf", "Result.pdf");

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputWordDocument =
assembly.GetManifestResourceStream("XamarinFormsAppl.Assets.EmployeesTemplat
e.docx");
//Loads an existing Word document into DocIO instance
WordDocument wordDocument = new WordDocument(inputWordDocument,
    FormatType.Automatic);
//document.Save(stream, FormatType.Docx);
DocIORenderer docIORenderer = new DocIORenderer();
PdfDocument pdfDocument = docIORenderer.ConvertToPDF(wordDocument);
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
pdfDocument.Close();
wordDocument.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.pdf",
    "application/pdf", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

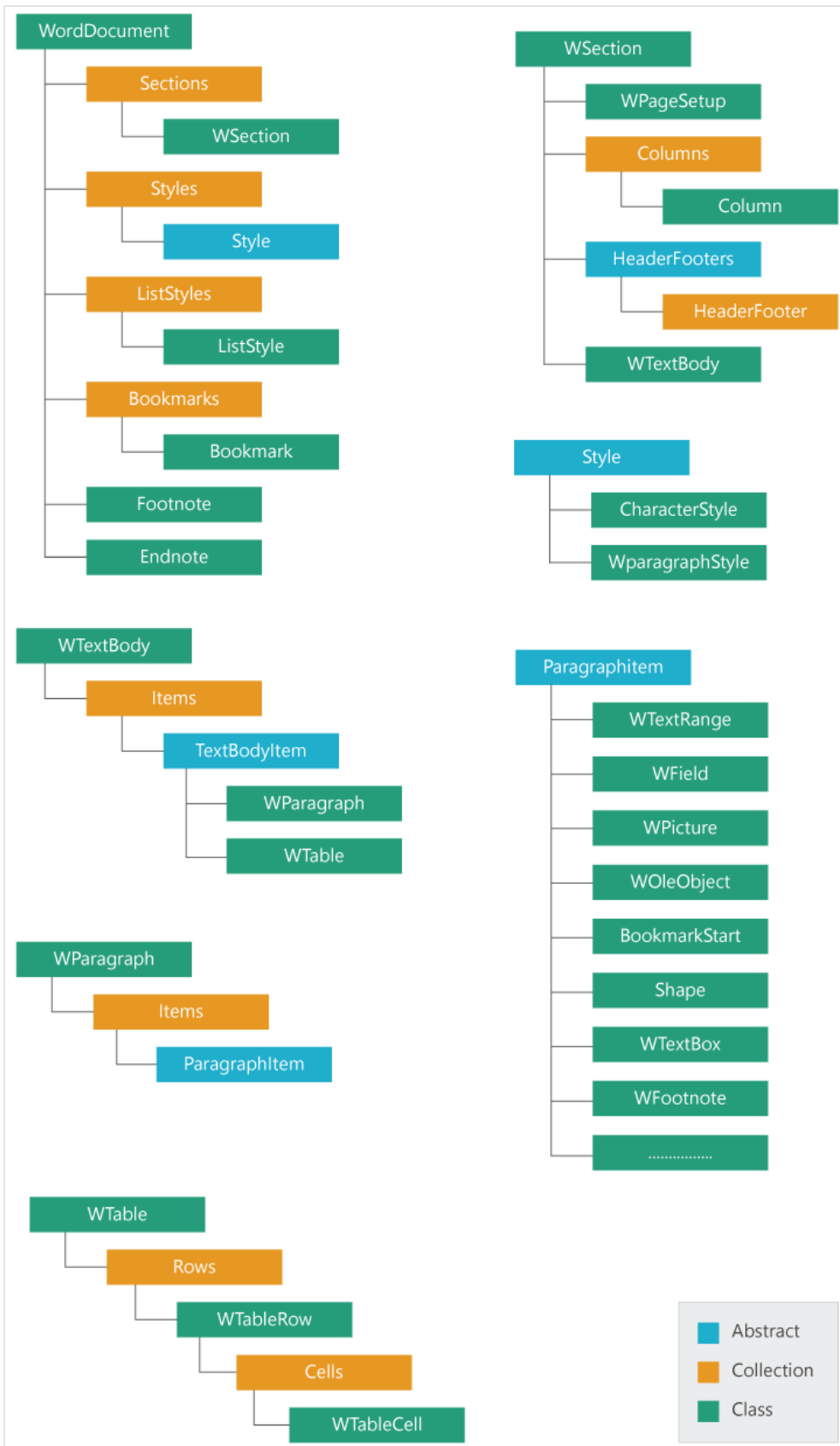
Note:

When the ChartToImageConverter object is not initialized, then the charts in Word document gets skipped during Word to PDF conversion.

ChartToImageConverter is supported from .NET Framework 4.0 onwards

Document Object Model representation

When an existing document is opened or a new document is created, the DocIO library creates a **Document Object Model (DOM)** of the document in main memory. This object model can be used to manipulate the document as needed.



Loading & saving document

Opening an existing document

You can open an existing Word document by using either the **Open** method or the constructor of **WordDocument** class

C#

```
//Opens an existing document from file system through constructor of WordDocument class
WordDocument document = new WordDocument(fileName);
```

VB.NET

```
'Opens an existing document from file system through constructor of WordDocument class
Dim document As New WordDocument(fileName)
```

UWP

```
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document through constructor of `WordDocument` class
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Tes
t.docx"), FormatType.Docx);
```

ASP.NET CORE

```
//Opens an existing document from stream through constructor of
`WordDocument` class
FileStream fileStreamPath = new FileStream(@"Data/Hello World.docx",
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath,
FormatType.Automatic);
```

XAMARIN

```
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document through constructor of `WordDocument` class
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("XamarinFormsAppl.Assets.Tes
t.docx"), FormatType.Automatic);
```

C#

```
//Creates an empty Word document instance
WordDocument document = new WordDocument();
//Loads or opens an existing word document through Open method of WordDocument class
document.Open(fileName);
```

VB.NET

```
'Creates an empty Word document instance
Dim document As New WordDocument()
'Loads or opens an existing word document through Open method of
WordDocument class
document.Open(fileName)
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".docx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
WordDocument document = new WordDocument();
await document.OpenAsync(inputStorageFile);
```

ASP.NET CORE

```
//Opens an existing document from stream through constructor of
WordDocument` class
FileStream fileStreamPath = new FileStream(@"Data/Hello World.docx",
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
//Creates an empty Word document instance
WordDocument document = new WordDocument();
//Loads or opens an existing word document through Open method of
WordDocument class
document.Open(fileStreamPath, FormatTypeAutomatic);
```

XAMARIN

```
/"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Creates an empty Word document instance
WordDocument document = new WordDocument();
//Loads or opens an existing word document through Open method of
WordDocument class
document.Open(assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Test.docx"), FormatTypeAutomatic);
```

Opening an existing document from Stream

You can open an existing document from stream by using either the overloads of **Open** methods or the constructor of **WordDocument** class

C#

```
//Opens an existing document from stream through constructor of WordDocument
class
WordDocument document = new WordDocument(wordDocumentStream,
FormatTypeAutomatic);
```

VB.NET

```
'Opens an existing document from stream through constructor of WordDocument
class
Dim document As New WordDocument(wordDocumentStream, FormatTypeAutomatic)
```

UWP

```
///"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
///Loads or opens an existing Word document from stream
Stream inputStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Test.docx");
///Opens an existing document through constructor of `WordDocument` class
WordDocument document = new WordDocument(inputStream, FormatType.Docx);
```

ASP.NET CORE

```
//Opens an existing document from stream through constructor of
`WordDocument` class
FileStream fileStreamPath = new FileStream(@"Data/Hello World.docx",
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
//Opens an existing document from stream through constructor of WordDocument
class
WordDocument document = new WordDocument(fileStreamPath,
FormatTypeAutomatic);
```

XAMARIN

```
///"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
///Loads or opens an existing Word document from stream
Stream inputStream =
assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Hello
World.docx");
///Opens an existing document through constructor of `WordDocument` class
WordDocument document = new WordDocument(inputStream, FormatTypeAutomatic);
```

C#

```
///Creates an empty WordDocument instance
WordDocument document = new WordDocument();
///Loads or opens an existing Word document through Open method of
WordDocument class
document.Open(wordDocumentStream, FormatTypeAutomatic);
```

VB.NET

```
'Creates an empty WordDocument instance
Dim document As New WordDocument()
'Loads or opens an existing word document through Open method of
WordDocument class
document.Open(wordDocumentStream, FormatTypeAutomatic)
```

UWP


```
// "App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
// Creates an empty WordDocument instance
using (WordDocument document = new WordDocument())
{
    // Loads or opens an existing Word document from stream
    Stream inputStream =
        assembly.GetManifestResourceStream("CreateWordSample.Assets.Test.docx");
    // Loads or opens an existing Word document through Open method of
    WordDocument class
    document.Open(inputStream, FormatType.Docx);
}
```

ASP.NET CORE

```
// Creates an empty WordDocument instance
using (WordDocument document = new WordDocument())
{
    // Loads or opens an existing Word document from stream
    FileStream fileStreamPath = new FileStream(@"Data/Hello World.docx",
        FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
    // Loads or opens an existing Word document through Open method of
    WordDocument class
    document.Open(fileStreamPath, FormatTypeAutomatic);
}
```

XAMARIN

```
// "App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
// Creates an empty WordDocument instance
using (WordDocument document = new WordDocument())
{
    // Loads or opens an existing Word document from stream
    Stream inputStream =
        assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Hello
        World.docx");
    // Loads or opens an existing Word document through Open method of
    WordDocument class
    document.Open(inputStream, FormatTypeAutomatic);
}
```

Opening an Encrypted Word document

You can open an existing encrypted Word document from either the file system or the stream by using the following overloads as shown.

C#

```
// Opens an existing encrypted document through constructor of WordDocument
class
WordDocument document = new WordDocument(fileName, FormatTypeAutomatic,
    "password");
```

VB.NET

```
'Opens an existing encrypted document through constructor of WordDocument class
Dim document As New WordDocument(fileName, FormatTypeAutomatic, "password")
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".docx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
WordDocument document = new WordDocument();
await document.OpenAsync(inputStorageFile, FormatType.Docx, "password");
```

ASP.NET CORE

```
//DocIO supports Encryption in Windows Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms alone.
```

XAMARIN

```
//DocIO supports Encryption in Windows Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms alone.
```

C#

```
//Creates an empty Word document instance
WordDocument document = new WordDocument();
//Loads or opens an existing encrypted Word document through Open method of WordDocument class
document.Open(wordDocumentStream, FormatTypeAutomatic, "password");
```

VB.NET

```
'Creates an empty Word document instance
Dim document As New WordDocument()
'Loads or opens an existing encrypted Word document through Open method of WordDocument class
document.Open(wordDocumentStream, FormatTypeAutomatic, "password")
```

UWP

```
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Creates an empty WordDocument instance
using (WordDocument document = new WordDocument())
{
    //Loads or opens an existing Word document from stream
    Stream inputStream =
    assembly.GetManifestResourceStream("CreateWordSample.Assets.Test.docx");
    //Loads or opens an existing encrypted Word document through Open method of WordDocument class
    document.Open(inputStream, FormatType.Docx, "password");
}
```

```
}
```

ASP.NET CORE

```
//DocIO supports Encryption in Windows Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms alone.
```

XAMARIN

```
//DocIO supports Encryption in Windows Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms alone.
```

Opening the read only Word document

You can open the ready only documents or read only streams using the OpenReadOnly method. If the Word document for reading is opened by any other application such as Microsoft Word, then the same document can be opened using DocIO in ReadOnly mode. The following code sample demonstrates the same.

C#

```
//Creates an empty WordDocument instance
WordDocument document = new WordDocument();
//Loads or opens an existing word document using read only stream
document.OpenReadOnly("Template.docx", Syncfusion.DocIO.FormatType.Docx);
```

VB.NET

```
'Creates an empty WordDocument instance
Dim document As WordDocument = New WordDocument
'Loads or opens an existing word document using read only stream
document.OpenReadOnly("Template.docx", Syncfusion.DocIO.FormatType.Docx)
```

UWP

```
//DocIO supports OpenReadOnly Word documents in Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms alone.
```

ASP.NET CORE

```
//DocIO supports OpenReadOnly Word documents in Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms alone.
```

XAMARIN

```
//DocIO supports OpenReadOnly Word documents in Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms alone.
```

You can also open an existing encrypted document in read only mode using the overloads as mentioned below.

C#

```
//Creates an empty WordDocument instance
WordDocument document = new WordDocument();
//Loads or opens an existing encrypted word document using read only stream
document.OpenReadOnly("Template.docx", Syncfusion.DocIO.FormatType.Docx,
"password");
```

VB.NET

```
'Creates an empty WordDocument instance
Dim document As WordDocument = New WordDocument
'Loads or opens an existing encrypted word document using read only stream
document.OpenReadOnly("Template.docx", Syncfusion.DocIO.FormatType.Docx,
"password")
```

UWP

```
//DocIO supports OpenReadOnly Word documents in Windows Forms, WPF, ASP.NET,
and ASP.NET MVC platforms alone.
```

ASP.NET CORE

```
//DocIO supports OpenReadOnly Word documents in Windows Forms, WPF, ASP.NET,
and ASP.NET MVC platforms alone.
```

XAMARIN

```
//DocIO supports OpenReadOnly Word documents in Windows Forms, WPF, ASP.NET,
and ASP.NET MVC platforms alone.
```

Saving a Word document to file system

You can save the created or manipulated Word document to file system using **Save** method of **WordDocument** class. When you do not provide the format type, then the document is saved in Word 97-2003 (*.doc) format.

C#

```
//Creates an empty WordDocument instance
WordDocument document = new WordDocument();
//opens an existing Word document through Open method of WordDocument class
document.Open(fileName);
//To-Do some manipulation
//To-Do some manipulation
//Saves the document in file system
document.Save(outputFileName, FormatType.Docx);
```

VB.NET

```
'Creates an empty WordDocument instance
Dim document As New WordDocument()
'opens an existing Word document through Open method of WordDocument class
document.Open(fileName)
'To-Do some manipulation
'To-Do some manipulation
```

```
'Saves the document in file system
document.Save(outputFileName, FormatType.Docx)
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".docx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
WordDocument document = new WordDocument();
await document.OpenAsync(inputStorageFile);
//To-Do some manipulation
//To-Do some manipulation
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = OutputFileName;
savePicker.FileTypeChoices.Add("Word Documents", new List<string>() {
".docx" });
//Creates a storage file from FileSavePicker
StorageFile outputStorageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await document.SaveAsAsync(outputStorageFile, FormatType.Docx);
```

ASP.NET CORE

```
//Open an existing WordDocument
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
WordDocument document = new WordDocument(inputStream, FormatType.Docx);
//To-Do some manipulation
//To-Do some manipulation
//Saving the Word document
FileStream outputStream = new FileStream("Sample.docx", FileMode.Create,
FileAccess.ReadWrite, FileShare.ReadWrite);
document.Save(outputStream, FormatType.Docx);
document.Close();
outputStream.Flush();
outputStream.Dispose();
```

XAMARIN

```
/"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(inputFilePath);
WordDocument document = new WordDocument(inputStream, FormatType.Docx);
//To-Do some manipulation
//To-Do some manipulation
//Saving the Word document
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
stream.Position = 0;
//Save the stream as a file in the device and invoke it for viewing
```

```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Closes the document
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Saving a Word document to Stream

You can also save the created or manipulated word document to stream by using overloads of **Save** methods

C#

```
//Creates an empty WordDocument instance
WordDocument document = new WordDocument();
//Opens an existing Word document through Open method of WordDocument class
document.Open(fileName);
//To-Do some manipulation
//To-Do some manipulation
//Creates an instance of memory stream
MemoryStream stream = new MemoryStream();
//Saves the document to stream
document.Save(stream, FormatType.Docx);
```

VB.NET

```
'Creates an empty WordDocument instance
Dim document As New WordDocument()
'Opens an existing Word document through Open method of WordDocument class
document.Open(fileName)
'To-Do some manipulation
'To-Do some manipulation
'Creates an instance of memory stream
Dim stream As New MemoryStream()
'Saves the document to stream
document.Save(stream, FormatType.Docx)
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".docx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
WordDocument document = new WordDocument();
await document.OpenAsync(inputStorageFile);
//To-Do some manipulation
//To-Do some manipulation
//Creates an instance of memory stream
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
```

```
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates an empty WordDocument instance
using (WordDocument document = new WordDocument())
{
    //Loads or opens an existing Word document from stream
    FileStream fileStreamPath = new FileStream(@"Data/Hello World.docx",
    FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
    //Loads or opens an existing Word document through Open method of
    WordDocument class
    document.Open(fileStreamPath, FormatType.Automatic);
    //To-Do some manipulation
    //To-Do some manipulation
    //Creates an instance of memory stream
    MemoryStream stream = new MemoryStream();
    //Saves the document to stream
    document.Save(stream, FormatType.Docx);
    //Closes the document
    document.Close();
    stream.Position = 0;
    //Download Word document in the browser
    return File(stream, "application/msword", "Result.docx");
}
```

XAMARIN

```
///"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
///Creates an empty WordDocument instance
using (WordDocument document = new WordDocument())
{
    ///Loads or opens an existing Word document from stream
    Stream inputStream =
    assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Hello
    World.docx");
    ///Loads or opens an existing Word document through Open method of
    WordDocument class
    document.Open(inputStream, FormatType.Automatic);
    ///To-Do some manipulation
    ///To-Do some manipulation
    ///Creates an instance of memory stream
    MemoryStream stream = new MemoryStream();
    ///Saves the document to stream
    document.Save(stream, FormatType.Docx);
    ///Closes the document
    document.Close();
    ///Save the stream as a file in the device and invoke it for viewing
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
    "application/msword", stream);
}
```

```
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Sending to a client browser

You can save and send the document to a client browser from a web site or web application by invoking the following shown overload of **Save** method. This method explicitly makes use of an instance of [HttpResponse](#) as its parameter in order to stream the document to client browser. So this overload is suitable for web application that references System.Web assembly.

C#

```
//Creates an empty WordDocument instance
WordDocument document = new WordDocument();
//Opens an existing Word document through Open method of WordDocument class
document.Open(fileName);
//To-Do some manipulation
//To-Do some manipulation
//Creates an instance of memory stream
MemoryStream stream = new MemoryStream();
//Saves the document to stream
document.Save(outputFileName, FormatType.Docx, Response,
HttpContentDisposition.Attachment);
```

VB.NET

```
'Creates an empty WordDocument instance
Dim document As New WordDocument()
'Opens an existing Word document through Open method of WordDocument class
document.Open(fileName)
'To-Do some manipulation
'To-Do some manipulation
'Creates an instance of memory stream
Dim stream As New MemoryStream()
'Saves the document to stream
document.Save(outputFileName, FormatType.Docx, Response,
HttpContentDisposition.Attachment)
```

UWP

```
//Saving and sending the Word document to a client browser from a web site
is suitable for web applications alone.
```

ASP.NET CORE

```
//Creates a new instance of WordDocument (Empty Word Document)
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to the created paragraph
```



```
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Saving and sending the Word document to a client browser from a web site
is suitable for web applications alone.
```

Closing a document

Once the document manipulation and save operation are completed, you should close the instance of **WordDocument**, in order to release all the memory consumed by DocIO's DOM. The following code example illustrates how to close a **WordDocument** instance.

C#

```
//Creates an empty WordDocument instance
WordDocument document = new WordDocument();
//opens an existing word document through Open method of WordDocument class
document.Open(fileName);
//To-Do some manipulation
//To-Do some manipulation
//Creates an instance of memory stream
MemoryStream stream = new MemoryStream();
//Saves the document to stream
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'creates an empty WordDocument instance
Dim document As New WordDocument()
'opens an existing word document through Open method of WordDocument class
document.Open(fileName)
'To-Do some manipulation
'To-Do some manipulation
'Creates an instance of memory stream
Dim stream As New MemoryStream()
'Saves the document to stream
document.Save(stream, FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
//Instantiates the File Picker
```

```

FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".docx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
WordDocument document = new WordDocument();
await document.OpenAsync(inputStorageFile);
//To-Do some manipulation
//To-Do some manipulation
//Creates an instance of memory stream
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an empty WordDocument instance
using (WordDocument document = new WordDocument())
{
    //Loads or opens an existing Word document from stream
    FileStream fileStreamPath = new FileStream(@"Data/Hello World.docx",
    FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
    //Loads or opens an existing Word document through Open method of
    WordDocument class
    document.Open(fileStreamPath, FormatTypeAutomatic);
    //To-Do some manipulation
    //To-Do some manipulation
    //Creates an instance of memory stream
    MemoryStream stream = new MemoryStream();
    //Saves the document to stream
    document.Save(stream, FormatType.Docx);
    //Closes the document
    document.Close();
    stream.Position = 0;
    //Download Word document in the browser
    return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

// "App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Creates an empty WordDocument instance
using (WordDocument document = new WordDocument())
{
    //Loads or opens an existing Word document from stream
    Stream inputStream =
    assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Hello
    World.docx");
}

```

```
//Loads or opens an existing Word document through Open method of
WordDocument class
document.Open(inputStream, FormatType.Automatic);
//To-Do some manipulation
//To-Do some manipulation
//Creates an instance of memory stream
MemoryStream stream = new MemoryStream();
//Saves the document to stream
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Closes the document
document.Close();
}
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Working with Word document

Iterating through document elements

The following are the important points to be remembered while iterating the document elements

- Document consists of one or more sections.
- Section contains the contents present in Headers, Footers and main document through the instances of **WTextBody**.
- **WTextBody** contains two type of elements – either paragraph or table

The following code example shows how to iterate throughout the Word document and remove the paragraph with a particular style.

C#

```
//Opens an existing document from file system through constructor of
WordDocument class
WordDocument document = new WordDocument(@"TestDocument.docx");
//Processes the body contents for each section in the Word document
foreach (WSection section in document.Sections)
{
    //Accesses the Body of section where all the contents in document are apart
    WTextBody sectionBody = section.Body;
    IterateTextBody(sectionBody);
    WHeadersFooters headersFooters = section.HeadersFooters;
    //Consider that OddHeader and OddFooter are applied to this document
    //Iterates through the TextBody of OddHeader and OddFooter
    IterateTextBody(headersFooters.OddHeader);
    IterateTextBody(headersFooters.OddFooter);
}
//Saves and closes the document instance
document.Save("Result.docx");
document.Close();
```

VB.NET

```

'Opens an existing document from file system through constructor of WordDocument class
Dim document As New WordDocument("TestDocument.docx")
'Processes the body contents for each section in the Word document
For Each section As WSection In document.Sections
'Accesses the Body of section where all the contents in document are apart
Dim sectionBody As WTextBody = section.Body
IterateTextBody(sectionBody)
Dim headersFooters As WHeadersFooters = section.HeadersFooters
'Consider that OddHeader and OddFooter are applied to this document
'Iterates through the text body of OddHeader and OddFooter
IterateTextBody(headersFooters.OddHeader)
IterateTextBody(headersFooters.OddFooter)
Next
'Saves and closes the document instance
document.Save("Result.docx")
document.Close()

```

UWP

```

///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document from file system through constructor of WordDocument class
using (WordDocument document = new
WordDocument((assembly.GetManifestResourceStream("CreateWordSample.Assets.Test.docx")), FormatType.Docx))
{
    foreach (WSection section in document.Sections)
    {
//Accesses the Body of section where all the contents in document are apart
WTextBody sectionBody = section.Body;
IterateTextBody(sectionBody);
WHeadersFooters headersFooters = section.HeadersFooters;
//Consider that OddHeader and OddFooter are applied to this document
//Iterates through the TextBody of OddHeader and OddFooter
IterateTextBody(headersFooters.OddHeader);
IterateTextBody(headersFooters.OddFooter);
    }
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
//Closes the Word document
document.Close();
}

```

ASP.NET CORE

```

FileStream fileStreamPath = new FileStream(@"Data>Hello World.docx",
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);

```

```
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(fileStreamPath,
FormatType.Automatic))
{
    foreach (WSection section in document.Sections)
    {
        //Accesses the Body of section where all the contents in document are apart
        WTextBody sectionBody = section.Body;
        IterateTextBody(sectionBody);
        WHeadersFooters headersFooters = section.HeadersFooters;
        //Consider that OddHeader and OddFooter are applied to this document
        //Iterates through the TextBody of OddHeader and OddFooter
        IterateTextBody(headersFooters.OddHeader);
        IterateTextBody(headersFooters.OddFooter);
    }
    MemoryStream stream = new MemoryStream();
    document.Save(stream, FormatType.Docx);
    //Closes the Word document
    document.Close();
    stream.Position = 0;
    //Download Word document in the browser
    return File(stream, "application/msword", "Result.docx");
}
```

XAMARIN

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new
WordDocument((assembly.GetManifestResourceStream("XamarinFormsAppl.Assets.He
llo World.docx")), FormatType.Automatic))
{
    foreach (WSection section in document.Sections)
    {
        //Accesses the Body of section where all the contents in document are apart
        WTextBody sectionBody = section.Body;
        IterateTextBody(sectionBody);
        WHeadersFooters headersFooters = section.HeadersFooters;
        //Consider that OddHeader and OddFooter are applied to this document
        //Iterates through the TextBody of OddHeader and OddFooter
        IterateTextBody(headersFooters.OddHeader);
        IterateTextBody(headersFooters.OddFooter);
    }
    MemoryStream stream = new MemoryStream();
    document.Save(stream, FormatType.Docx);
    //Save the stream as a file in the device and invoke it for viewing
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
    //Closes the Word document
    document.Close();
    //Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

```
}

```

The following code example provides supporting methods for the above code.

C#

```
private static void IterateTextBody(WTextBody textBody)
{
    //Iterates through each of the child items of WTextBody
    for (int i = 0; i < textBody.ChildEntities.Count; i++)
    {
        //IEntity is the basic unit in DocIO DOM.
        //Accesses the body items (should be either paragraph or table) as IEntity
        IEntity bodyItemEntity = textBody.ChildEntities[i];
        //A Text body has 2 types of elements - Paragraph and Table
        //Decides the element type by using EntityType
        switch (bodyItemEntity.EntityType)
        {
            case EntityType.Paragraph:
                WParagraph paragraph = bodyItemEntity as WParagraph;
                //Checks for particular style name and removes the paragraph from DOM
                if (paragraph.StyleName == "MyStyle")
                {
                    int index = textBody.ChildEntities.IndexOf(paragraph);
                    textBody.ChildEntities.RemoveAt(index);
                }
                break;
            case EntityType.Table:
                //Table is a collection of rows and cells
                //Iterates through table's DOM
                IterateTable(bodyItemEntity as WTable);
                break;
        }
    }
}
```

VB.NET

```
Private Shared Sub IterateTextBody(textBody As WTextBody)
    'Iterates through the each of the child items of WTextBody
    For i As Integer = 0 To textBody.ChildEntities.Count - 1
        'IEntity is the basic unit in DocIO DOM.
        'Accesses the body items (should be either paragraph or table) as IEntity
        Dim bodyItemEntity As IEntity = textBody.ChildEntities(i)
        'A Text body has 2 types of elements - Paragraph and Table
        'decide the element type using EntityType
        Select Case bodyItemEntity.EntityType
            Case EntityType.Paragraph
                Dim paragraph As WParagraph = TryCast(bodyItemEntity, WParagraph)
                'Checks for a particular style name and removes the paragraph from DOM
                If paragraph.StyleName = "MyStyle" Then
                    Dim index As Integer = textBody.ChildEntities.IndexOf(paragraph)
                    textBody.ChildEntities.RemoveAt(index)
                End If
            Exit Select
            Case EntityType.Table
```

```

'Table is a collection of rows and cells
'Iterates through table's DOM
IterateTable(TryCast(bodyItemEntity, WTable))
Exit Select
End Select
Next
End Sub

```

UWP

```

async void IterateTextBody(WTextBody textBody)
{
    //Iterates through each of the child items of WTextBody
    for (int i = 0; i < textBody.ChildEntities.Count; i++)
    {
        //IEntity is the basic unit in DocIO DOM.
        //Accesses the body items (should be either paragraph or table) as IEntity
        IEntity bodyItemEntity = textBody.ChildEntities[i];
        //A Text body has 2 types of elements - Paragraph and Table
        //Decides the element type by using EntityType
        switch (bodyItemEntity.EntityType)
        {
            case EntityType.Paragraph:
                WParagraph paragraph = bodyItemEntity as WParagraph;
                //Checks for particular style name and removes the paragraph from DOM
                if (paragraph.StyleName == "MyStyle")
                {
                    int index = textBody.ChildEntities.IndexOf(paragraph);
                    textBody.ChildEntities.RemoveAt(index);
                }
                break;
            case EntityType.Table:
                //Table is a collection of rows and cells
                //Iterates through table's DOM
                IterateTable(bodyItemEntity as WTable);
                break;
        }
    }
}

```

ASP.NET CORE

```

private static void IterateTextBody(WTextBody textBody)
{
    //Iterates through each of the child items of WTextBody
    for (int i = 0; i < textBody.ChildEntities.Count; i++)
    {
        //IEntity is the basic unit in DocIO DOM.
        //Accesses the body items (should be either paragraph or table) as IEntity
        IEntity bodyItemEntity = textBody.ChildEntities[i];
        //A Text body has 2 types of elements - Paragraph and Table
        //Decides the element type by using EntityType
        switch (bodyItemEntity.EntityType)
        {
            case EntityType.Paragraph:
                WParagraph paragraph = bodyItemEntity as WParagraph;

```

```
//Checks for particular style name and removes the paragraph from DOM
if (paragraph.StyleName == "MyStyle")
{
    int index = textBody.ChildEntities.IndexOf(paragraph);
    textBody.ChildEntities.RemoveAt(index);
}
break;
case EntityType.Table:
//Table is a collection of rows and cells
//Iterates through table's DOM
IterateTable(bodyItemEntity as WTable);
break;
}
}
}
```

XAMARIN

```
private static void IterateTextBody(WTextBody textBody)
{
    //Iterates through each of the child items of WTextBody
    for (int i = 0; i < textBody.ChildEntities.Count; i++)
    {
        //IEntity is the basic unit in DocIO DOM.
        //Accesses the body items (should be either paragraph or table) as IEntity
        IEntity bodyItemEntity = textBody.ChildEntities[i];
        //A Text body has 2 types of elements - Paragraph and Table
        //Decides the element type by using EntityType
        switch (bodyItemEntity.EntityType)
        {
            case EntityType.Paragraph:
                WParagraph paragraph = bodyItemEntity as WParagraph;
                //Checks for particular style name and removes the paragraph from DOM
                if (paragraph.StyleName == "MyStyle")
                {
                    int index = textBody.ChildEntities.IndexOf(paragraph);
                    textBody.ChildEntities.RemoveAt(index);
                }
                break;
            case EntityType.Table:
                //Table is a collection of rows and cells
                //Iterates through table's DOM
                IterateTable(bodyItemEntity as WTable);
                break;
        }
    }
}
```

The following code example provides supporting methods for the above code.

C#

```
private static void IterateTable(WTable table)
{
    //Iterates the row collection in a table
    foreach (WTableRow row in table.Rows)
```



```

{
    //Iterates the cell collection in a table row
    foreach (WTableCell cell in row.Cells)
    {
        //Table cell is derived from (also a) TextBody
        //Reusing the code meant for iterating TextBody
        IterateTextBody(cell);
    }
}

```

VB.NET

```

Private Shared Sub IterateTable(table As WTable)
    'Iterates the row collection in a table
    For Each row As WTableRow In table.Rows
        'Iterates the cell collection in a table row
        For Each cell As WTableCell In row.Cells
            'Table cell is derived from (also a) TextBody
            'Reusing the code meant for iterating TextBody
            IterateTextBody(cell)
        Next
    Next
End Sub

```

UWP

```

async void IterateTable(WTable table)
{
    //Iterates the row collection in a table
    foreach (WTableRow row in table.Rows)
    {
        //Iterates the cell collection in a table row
        foreach (WTableCell cell in row.Cells)
        {
            //Table cell is derived from (also a) TextBody
            //Reusing the code meant for iterating TextBody
            IterateTextBody(cell);
        }
    }
}

```

ASP.NET CORE

```

private static void IterateTable(WTable table)
{
    //Iterates the row collection in a table
    foreach (WTableRow row in table.Rows)
    {
        //Iterates the cell collection in a table row
        foreach (WTableCell cell in row.Cells)
        {
            //Table cell is derived from (also a) TextBody
            //Reusing the code meant for iterating TextBody
            IterateTextBody(cell);
        }
    }
}

```

```
}
}
}
```

XAMARIN

```
private static void IterateTable(WTable table)
{
    //Iterates the row collection in a table
    foreach (WTableRow row in table.Rows)
    {
        //Iterates the cell collection in a table row
        foreach (WTableCell cell in row.Cells)
        {
            //Table cell is derived from (also a) TextBody
            //Reusing the code meant for iterating TextBody
            IterateTextBody(cell);
        }
    }
}
```

The following code example shows how to iterate throughout the paragraph and modify the hyperlink (Hyperlink)Uri and specific text (WTextRange)with another.

C#

```
//Opens an existing document from file system through constructor of
WordDocument class
WordDocument document = new WordDocument(@"TestDocument.docx");
//Processes the body contents for each section in the Word document
foreach (WSection section in document.Sections)
{
    //Accesses the Body of section where all the contents in document are apart
    WTextBody sectionBody = section.Body;
    IterateTextBody(sectionBody);
    WHeadersFooters headersFooters = section.HeadersFooters;
    //consider that OddHeader & OddFooter are applied to this document
    //Iterates through the TextBody of OddHeader and OddFooter
    IterateTextBody(headersFooters.OddHeader);
    IterateTextBody(headersFooters.OddFooter);
}
//Saves and closes the document instance
document.Save("Result.docx");
document.Close();
```

VB.NET

```
Dim document As New WordDocument("TestDocument.docx")
'Processes the body contents for each section in the Word document
For Each section As WSection In document.Sections
'Accesses the Body of section where all the contents in document are apart
Dim sectionBody As WTextBody = section.Body
IterateTextBody(sectionBody)
Dim headersFooters As WHeadersFooters = section.HeadersFooters
'Considers that OddHeader and OddFooter are applied to this document
```

```
'Iterates through the TextBody of OddHeader and
OddFooterIterateTextBody(headersFooters.OddHeader)
IterateTextBody(headersFooters.OddFooter)
Next
'Saves and closes the document instance
document.Save("Result.docx")
document.Close()
```

UWP

```
///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
///Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new
WordDocument((assembly.GetManifestResourceStream("CreateWordSample.Assets.Te
st.docx")), FormatType.Docx))
{
    foreach (WSection section in document.Sections)
    {
        ///Accesses the Body of section where all the contents in document are apart
        WTextBody sectionBody = section.Body;
        IterateTextBody(sectionBody);
        WHeadersFooters headersFooters = section.HeadersFooters;
        ///Consider that OddHeader and OddFooter are applied to this document
        ///Iterates through the TextBody of OddHeader and OddFooter
        IterateTextBody(headersFooters.OddHeader);
        IterateTextBody(headersFooters.OddFooter);
    }
    MemoryStream stream = new MemoryStream();
    await document.SaveAsync(stream, FormatType.Docx);
    ///Saves the stream as Word file in local machine
    Save(stream, "Result.docx");
    ///Please refer the below link to save Word document in UWP platform
    ///https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
    ///Closes the Word document
    document.Close();
}
```

ASP.NET CORE

```
FileStream fileStreamPath = new FileStream(@"Data/Hello World.docx",
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
///Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(fileStreamPath,
FormatType.Automatic))
{
    foreach (WSection section in document.Sections)
    {
        ///Accesses the Body of section where all the contents in document are apart
        WTextBody sectionBody = section.Body;
        IterateTextBody(sectionBody);
        WHeadersFooters headersFooters = section.HeadersFooters;
        ///Consider that OddHeader and OddFooter are applied to this document
```

```
//Iterates through the TextBody of OddHeader and OddFooter
IterateTextBody(headersFooters.OddHeader);
IterateTextBody(headersFooters.OddFooter);
}
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the Word document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}
```

XAMARIN

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new
WordDocument((assembly.GetManifestResourceStream("XamarinFormsAppl.Assets.He
llo World.docx")), FormatType.Automatic))
{
foreach (WSection section in document.Sections)
{
//Accesses the Body of section where all the contents in document are apart
WTextBody sectionBody = section.Body;
IterateTextBody(sectionBody);
WHeadersFooters headersFooters = section.HeadersFooters;
//Consider that OddHeader and OddFooter are applied to this document
//Iterates through the TextBody of OddHeader and OddFooter
IterateTextBody(headersFooters.OddHeader);
IterateTextBody(headersFooters.OddFooter);
}
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Closes the Word document
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
}
```

The following code example provides supporting methods for the above code.

C#

```
private static void IterateTextBody(WTextBody textBody)
{
//Iterates through each of the child items of WTextBody
for (int i = 0; i < textBody.ChildEntities.Count; i++)
{
//IEntity is the basic unit in DocIO DOM.
}
```

```

//Accesses the body items (should be either paragraph or table) as IEntity
IEntity bodyItemEntity = textBody.ChildEntities[i];
//A Text body has 2 types of elements - Paragraph and Table
//Decides the element type by using EntityType
switch (bodyItemEntity.EntityType)
{
    case EntityType.Paragraph:
        WParagraph paragraph = bodyItemEntity as WParagraph;
        //Processes the paragraph contents
        //Iterates through the paragraph's DOM
        IterateParagraph(paragraph);
        break;
    case EntityType.Table:
        //Table is a collection of rows and cells
        //Iterates through table's DOM
        IterateTable(bodyItemEntity as WTable);
        break;
}
}
}

```

VB.NET

```

Private Shared Sub IterateTextBody(textBody As WTextBody)
    'Iterates through each of the child items of WTextBody
    For i As Integer = 0 To textBody.ChildEntities.Count - 1
        'IEntity is the basic unit in DocIO DOM.
        'Accesses the body items (should be either paragraph or table) as IEntity
        Dim bodyItemEntity As IEntity = textBody.ChildEntities(i)
        'A Text body has 2 types of elements - Paragraph and Table
        'Decides the element type by using EntityType
        Select Case bodyItemEntity.EntityType
            Case EntityType.Paragraph
                Dim paragraph As WParagraph = TryCast(bodyItemEntity, WParagraph)
                'Processes the paragraph contents
                'Iterates through the paragraph's DOM
                IterateParagraph(paragraph)
            Exit Select
            Case EntityType.Table
                'Table is a collection of rows and cells
                'Iterates through table's DOM
                IterateTable(TryCast(bodyItemEntity, WTable))
            Exit Select
        End Select
    Next
End Sub

```

UWP

```

async void IterateTextBody(WTextBody textBody)
{
    //Iterates through each of the child items of WTextBody
    for (int i = 0; i < textBody.ChildEntities.Count; i++)
    {
        //IEntity is the basic unit in DocIO DOM.
        //Accesses the body items (should be either paragraph or table) as IEntity

```

```

IEntity bodyItemEntity = textBody.ChildEntities[i];
//A Text body has 2 types of elements - Paragraph and Table
//Decides the element type by using EntityType
switch (bodyItemEntity.EntityType)
{
case EntityType.Paragraph:
WParagraph paragraph = bodyItemEntity as WParagraph;
//Processes the paragraph contents
//Iterates through the paragraph's DOM
IterateParagraph(paragraph);
break;
case EntityType.Table:
//Table is a collection of rows and cells
//Iterates through table's DOM
IterateTable(bodyItemEntity as WTable);
break;
}
}
}

```

ASP.NET CORE

```

private static void IterateTextBody(WTextBody textBody)
{
//Iterates through each of the child items of WTextBody
for (int i = 0; i < textBody.ChildEntities.Count; i++)
{
//IEntity is the basic unit in DocIO DOM.
//Accesses the body items (should be either paragraph or table) as IEntity
IEntity bodyItemEntity = textBody.ChildEntities[i];
//A Text body has 2 types of elements - Paragraph and Table
//Decides the element type by using EntityType
switch (bodyItemEntity.EntityType)
{
case EntityType.Paragraph:
WParagraph paragraph = bodyItemEntity as WParagraph;
//Processes the paragraph contents
//Iterates through the paragraph's DOM
IterateParagraph(paragraph);
break;
case EntityType.Table:
//Table is a collection of rows and cells
//Iterates through table's DOM
IterateTable(bodyItemEntity as WTable);
break;
}
}
}

```

XAMARIN

```

private static void IterateTextBody(WTextBody textBody)
{
//Iterates through each of the child items of WTextBody
for (int i = 0; i < textBody.ChildEntities.Count; i++)
{

```

```

//IEntity is the basic unit in DocIO DOM.
//Accesses the body items (should be either paragraph or table) as IEntity
IEntity bodyItemEntity = textBody.ChildEntities[i];
//A Text body has 2 types of elements - Paragraph and Table
//Decides the element type by using EntityType
switch (bodyItemEntity.EntityType)
{
    case EntityType.Paragraph:
        WParagraph paragraph = bodyItemEntity as WParagraph;
        //Processes the paragraph contents
        //Iterates through the paragraph's DOM
        IterateParagraph(paragraph);
        break;
    case EntityType.Table:
        //Table is a collection of rows and cells
        //Iterates through table's DOM
        IterateTable(bodyItemEntity as WTable);
        break;
}
}
}

```

The following code example provides supporting methods for the above code.

C#

```

private static void IterateTable(WTable table)
{
    //Iterates the row collection in a table
    foreach (WTableRow row in table.Rows)
    {
        //Iterates the cell collection in a table row
        foreach (WTableCell cell in row.Cells)
        {
            //Table cell is derived from (also a) TextBody
            //Reusing the code meant for iterating TextBody
            IterateTextBody(cell);
        }
    }
}

```

VB.NET

```

Private Shared Sub IterateTable(table As WTable)
    'Iterates the row collection in a table
    For Each row As WTableRow In table.Rows
        'Iterates the cell collection in a table row
        For Each cell As WTableCell In row.Cells
            'Table cell is derived from (also a) TextBody
            'Reusing the code meant for iterating TextBody
            IterateTextBody(cell)
        Next
    Next
End Sub

```

UWP

```

async void IterateTable(WTable table)
{
    //Iterates the row collection in a table
    foreach (WTableRow row in table.Rows)
    {
        //Iterates the cell collection in a table row
        foreach (WTableCell cell in row.Cells)
        {
            //Table cell is derived from (also a) TextBody
            //Reusing the code meant for iterating TextBody
            IterateTextBody(cell);
        }
    }
}

```

ASP.NET CORE

```

private static void IterateTable(WTable table)
{
    //Iterates the row collection in a table
    foreach (WTableRow row in table.Rows)
    {
        //Iterates the cell collection in a table row
        foreach (WTableCell cell in row.Cells)
        {
            //Table cell is derived from (also a) TextBody
            //Reusing the code meant for iterating TextBody
            IterateTextBody(cell);
        }
    }
}

```

XAMARIN

```

private static void IterateTable(WTable table)
{
    //Iterates the row collection in a table
    foreach (WTableRow row in table.Rows)
    {
        //Iterates the cell collection in a table row
        foreach (WTableCell cell in row.Cells)
        {
            //Table cell is derived from (also a) TextBody
            //Reusing the code meant for iterating TextBody
            IterateTextBody(cell);
        }
    }
}

```

The following code example provides supporting methods for the above code.

C#

```

private static void IterateParagraph(WParagraph paragraph)

```



```

{
    for (int i = 0; i < paragraph.ChildEntities.Count; i++)
    {
        Entity entity = paragraph.ChildEntities[i];
        //A paragraph can have child elements such as text, image, hyperlink,
        symbols, etc.,
        //Decides the element type by using EntityType
        switch (entity.EntityType)
        {
            case EntityType.TextRange:
                //Replaces the text with another
                WTextRange textRange = entity as WTextRange;
                if (textRange.Text == "Andrew")
                {
                    (entity as WTextRange).Text = "Fuller";
                }
                break;
            case EntityType.Field:
                WField field = entity as WField;
                if (field.FieldType == FieldType.FieldHyperlink)
                {
                    //Creates hyperlink instance from field to manipulate the hyperlink
                    Hyperlink hyperlink = new Hyperlink(entity as WField);
                    //Modifies the Uri of the hyperlink
                    if (hyperlink.Type == HyperlinkType.WebLink && hyperlink.TextToDisplay ==
                        "HTML")
                    {
                        hyperlink.Uri = "http://www.w3schools.com/";
                    }
                }
                break;
        }
    }
}

```

VB.NET

```

Private Shared Sub IterateParagraph(paragraph As WParagraph)
    For i As Integer = 0 To paragraph.ChildEntities.Count - 1
        Dim entity As Entity = paragraph.ChildEntities(i)
        'A Paragraph has child elements such as text, image, hyperlink, symbols,
        etc.,
        'Decides the element type by using EntityType
        Select Case entity.EntityType
            Case EntityType.TextRange
                'Replaces the text with another
                Dim textRange As WTextRange = TryCast(entity, WTextRange)
                If textRange.Text = "Andrew" Then
                    TryCast(entity, WTextRange).Text = "Fuller"
                End If
            Exit Select
            Case EntityType.Field
                Dim field As WField = TryCast(entity, WField)
                If field.FieldType = FieldType.FieldHyperlink Then
                    'Creates Hyperlink instance from field to manipulate the Hyperlink
                    Dim hyperlink As New Hyperlink(TryCast(entity, WField))
                End If
        End Select
    Next
End Sub

```

```

'Modifies the Uri of the hyperlink
If hyperlink.Type = HyperlinkType.WebLink AndAlso hyperlink.TextToDisplay =
"HTML" Then
    hyperlink.Uri = "http://www.w3schools.com/"
End If
End If
Exit Select
End Select
Next
End Sub

```

UWP

```

async void IterateParagraph(WParagraph paragraph)
{
    for (int i = 0; i < paragraph.ChildEntities.Count; i++)
    {
        Entity entity = paragraph.ChildEntities[i];
        //A paragraph can have child elements such as text, image, hyperlink,
        symbols, etc.,
        //Decides the element type by using EntityType
        switch (entity.EntityType)
        {
            case EntityType.TextRange:
                //Replaces the text with another
                WTextRange textRange = entity as WTextRange;
                if (textRange.Text == "Andrew")
                {
                    (entity as WTextRange).Text = "Fuller";
                }
                break;
            case EntityType.Field:
                WField field = entity as WField;
                if (field.FieldType == FieldType.FieldHyperlink)
                {
                    //Creates hyperlink instance from field to manipulate the hyperlink
                    Hyperlink hyperlink = new Hyperlink(entity as WField);
                    //Modifies the Uri of the hyperlink
                    if (hyperlink.Type == HyperlinkType.WebLink && hyperlink.TextToDisplay ==
                        "HTML")
                    {
                        hyperlink.Uri = "http://www.w3schools.com/";
                    }
                }
                break;
        }
    }
}

```

ASP.NET CORE

```

private static void IterateParagraph(WParagraph paragraph)
{
    for (int i = 0; i < paragraph.ChildEntities.Count; i++)
    {
        Entity entity = paragraph.ChildEntities[i];
    }
}

```

```
//A paragraph can have child elements such as text, image, hyperlink,
symbols, etc.,
//Decides the element type by using EntityType
switch (entity.EntityType)
{
case EntityType.TextRange:
//Replaces the text with another
WTextRange textRange = entity as WTextRange;
if (textRange.Text == "Andrew")
{
(entity as WTextRange).Text = "Fuller";
}
break;
case EntityType.Field:
WField field = entity as WField;
if (field.FieldType == FieldType.FieldHyperlink)
{
//Creates hyperlink instance from field to manipulate the hyperlink
Hyperlink hyperlink = new Hyperlink(entity as WField);
//Modifies the Uri of the hyperlink
if (hyperlink.Type == HyperlinkType.WebLink && hyperlink.TextToDisplay ==
"HTML")
{
hyperlink.Uri = "http://www.w3schools.com/";
}
}
break;
}
}
```

XAMARIN

```
private static void IterateParagraph(WParagraph paragraph)
{
for (int i = 0; i < paragraph.ChildEntities.Count; i++)
{
Entity entity = paragraph.ChildEntities[i];
//A paragraph can have child elements such as text, image, hyperlink,
symbols, etc.,
//Decides the element type by using EntityType
switch (entity.EntityType)
{
case EntityType.TextRange:
//Replaces the text with another
WTextRange textRange = entity as WTextRange;
if (textRange.Text == "Andrew")
{
(entity as WTextRange).Text = "Fuller";
}
break;
case EntityType.Field:
WField field = entity as WField;
if (field.FieldType == FieldType.FieldHyperlink)
{
//Creates hyperlink instance from field to manipulate the hyperlink
```

```

Hyperlink hyperlink = new Hyperlink(entity as WField);
//Modifies the Uri of the hyperlink
if (hyperlink.Type == HyperlinkType.WebLink && hyperlink.TextToDisplay ==
"HTML")
{
hyperlink.Uri = "http://www.w3schools.com/";
}
}
break;
}
}
}

```

Cloning a Word document

You can create a deep copy of a Word document by using `Clone` method of `WordDocument` class. You can read the template document from file system or stream and create multiple document copies by cloning it. This improves the performance of document generation, as there is no need to read the Word document each time.

C#

```

//Opens an existing document
WordDocument inputTemplateDoc = new WordDocument(fileName);
//Creates a clone of Input Template
WordDocument clonedDocument = inputTemplateDoc.Clone();
//Saves and closes the cloned document instance
clonedDocument.Save("ClonedDocument.docx");
clonedDocument.Close();
//Closes the input template document instance
sourceDocument.Close();

```

VB.NET

```

'Opens an existing document
Dim inputTemplateDoc As New WordDocument(fileName)
'Creates a clone of Input Template
Dim clonedDocument As WordDocument = inputTemplateDoc.Clone()
'Saves and closes the cloned document instance
clonedDocument.Save("ClonedDocument.docx")
clonedDocument.Close()
'Closes the input template document instance
sourceDocument.Close()

```

UWP

```

//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Tes
t.docx"), FormatType.Docx))
{
WordDocument clonedDocument = document.Clone();
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream

```

```
await clonedDocument.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
document.Close();
clonedDocument.Close();
}
```

ASP.NET CORE

```
FileStream fileStreamPath = new FileStream(@"Data/Hello World.docx",
    FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(fileStreamPath,
    FormatTypeAutomatic))
{
    //Creates a clone of Input Template
    WordDocument clonedDocument = document.Clone();
    MemoryStream stream = new MemoryStream();
    //Saves and closes the cloned document instance
    clonedDocument.Save(stream, FormatType.Docx);
    //Closes the document
    document.Close();
    clonedDocument.Close();
    stream.Position = 0;
    //Download Word document in the browser
    return File(stream, "application/msword", "Result.docx");
}
```

XAMARIN

```
///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
    WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Tes
t.docx"), FormatTypeAutomatic))
{
    WordDocument clonedDocument = document.Clone();
    MemoryStream stream = new MemoryStream();
    clonedDocument.Save(stream, FormatType.Docx);
    //Save the stream as a file in the device and invoke it for viewing
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWordDoc.doc
x", "application/msword", stream);
    //Closes the document
    clonedDocument.Close();
    document.Close();
    //Please download the helper files from the below link to save the stream as
    file and open the file for viewing in Xamarin platform
    //https://help.syncfusion.com/file-formats/docio/create-word-document-in-
    xamarin#helper-files-for-xamarin
}
```

You can also create a deep copy of document elements such as sections, paragraphs, Tables, Text, Image, OleObject, Shapes, TextBoxes and etc., The following code example illustrates how to clone the section and save each cloned section as a Word document.

C#

```
//Opens a source document
WordDocument sourceDocument = new WordDocument("SourceDocument.docx");
//Processes the each section in the Word document
for (int i = 0; i < sourceDocument.Sections.Count;i++)
{
    //Creates new WordDocument instance to add cloned section
    WordDocument destinationDocument = new WordDocument();
    //Clones and adds source document sections to the destination document
    destinationDocument.Sections.Add(sourceDocument.Sections[i].Clone());
    //Saves and closes the document instance
    destinationDocument.Save("Section_" + i + ".docx");
    destinationDocument.Close();
}
//Closes the source document instance
sourceDocument.Close();
```

VB.NET

```
'Opens a source document
Dim sourceDocument As New WordDocument("SourceDocument.docx")
'Processes the each section in the Word document
For i As Integer = 0 To sourceDocument.Sections.Count - 1
    'Creates new WordDocument instance to add cloned section
    Dim destinationDocument As New WordDocument()
    'Clones and adds source document sections to the destination document
    destinationDocument.Sections.Add(sourceDocument.Sections(i).Clone())
    'Saves and closes the document instance
    destinationDocument.Save("Section_" + i + ".docx")
    destinationDocument.Close()
Next
'Closes the source document instance
sourceDocument.Close()
```

UWP

```
//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument sourceDocument = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.SourceDocumen
t.docx"), FormatType.Docx);
//Processes the each section in the Word document
for (int i = 0; i < sourceDocument.Sections.Count;i++)
{
    //Creates new WordDocument instance to add cloned section
    WordDocument destinationDocument = new WordDocument();
    //Clones and adds source document sections to the destination document
    destinationDocument.Sections.Add(sourceDocument.Sections[i].Clone());
    //Saves the Word file to MemoryStream
    MemoryStream stream = new MemoryStream();
    await destinationDocument.SaveAsync(stream, FormatType.Docx);
}
```

```
//Saves the stream as Word file in local machine.Please find Save method in
[link] (https://help.syncfusion.com/file-formats/docio/create-word-document-
in-uwp#save-word-document-in-uwp)
Save(stream, "Section_" + i + ".docx");
destinationDocument.Close();
}
//Closes the source document instance
sourceDocument.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates an instance of WordDocument class
FileStream fileStreamPath = new FileStream("SourceDocument.docx",
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
WordDocument sourceDocument = new WordDocument(fileStreamPath);
//Processes the each section in the Word document
for (int i = 0; i < sourceDocument.Sections.Count; i++)
{
//Creates new WordDocument instance to add cloned section
WordDocument destinationDocument = new WordDocument();
//Clones and adds source document sections to the destination document
destinationDocument.Sections.Add(sourceDocument.Sections[i].Clone());
//Saves and closes the document instance
FileStream outputStream = new FileStream("Section_" + i + ".docx",
FileMode.Create, FileAccess.ReadWrite, FileShare.ReadWrite);
destinationDocument.Save(outputStream, FormatType.Docx);
destinationDocument.Close();
outputStream.Flush();
outputStream.Dispose();
}
//Closes the source document instance
sourceDocument.Close();
```

XAMARIN

```
//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument sourceDocument = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Assets.Sourc
eDocument.docx"), FormatType.Docx);
//Processes the each section in the Word document
for (int i = 0; i < sourceDocument.Sections.Count; i++)
{
//Creates new WordDocument instance to add cloned section
WordDocument destinationDocument = new WordDocument();
//Clones and adds source document sections to the destination document
destinationDocument.Sections.Add(sourceDocument.Sections[i].Clone());
//Saves and closes the document instance
MemoryStream stream = new MemoryStream();
destinationDocument.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Section_" + i +
".docx", "application/msword", stream);
```

```
destinationDocument.Close();
//Please download the helper files from the below link to save the stream as
//file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
//xamarin#helper-files-for-xamarin
}
//Closes the source document instance
sourceDocument.Close();
```

Merging Word documents

You can merge multiple Word documents into single Word document by using DocIO's capability of importing contents from one document to another. The imported contents are appended at the end of document. The following code example illustrates how to import the contents from source document into destination document where the contents are appended.

C#

```
//Opens the source document
WordDocument sourceDocument = new WordDocument(sourceFileName);
//Opens the destination document
WordDocument destinationDocument = new WordDocument(targetFileName);
//Imports the contents of source document at the end of destination document
destinationDocument.ImportContent(sourceDocument,
ImportOptions.UseDestinationStyles);
//Saves the destination document
destinationDocument.Save(outputFileName, FormatType.Docx);
//closes the document instances
sourceDocument.Close();
destinationDocument.Close();
```

VB.NET

```
'Opens the source document
Dim sourceDocument As New WordDocument(sourceFileName)
'Opens the destination document
Dim destinationDocument As New WordDocument(targetFileName)
'Imports the contents of source document at the end of destination document
destinationDocument.ImportContent(sourceDocument,
ImportOptions.UseDestinationStyles)
'Saves the destination document
destinationDocument.Save(outputFileName, FormatType.Docx)
'closes the document instances
sourceDocument.Close()
destinationDocument.Close()
```

UWP

```
///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Tes
t.docx"), FormatType.Docx))
{
    //Opens the destination document
```



```

WordDocument destinationDocument = new
WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Letter
Formatting.docx"), FormatType.Docx);
//Imports the contents of source document at the end of destination document
destinationDocument.ImportContent(document,
ImportOptions.UseDestinationStyles);
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await destinationDocument.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine. Please find Save method in
[link](https://help.syncfusion.com/file-formats/docio/create-word-document-
in-uwp#save-word-document-in-uwp)
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
document.Close();
destinationDocument.Close();
}

```

ASP.NET CORE

```

FileStream sourceStreamPath = new FileStream(sourceFileName, FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
FileStream destinationStreamPath = new FileStream(destinationFileName,
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
//Opens an source document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(sourceStreamPath,
FormatType.Automatic))
{
//Opens the destination document
WordDocument destinationDocument = new WordDocument(destinationStreamPath,
FormatType.Docx);
//Imports the contents of source document at the end of destination document
destinationDocument.ImportContent(document,
ImportOptions.UseDestinationStyles);
MemoryStream stream = new MemoryStream();
//Saves and closes the destination document to MemoryStream
destinationDocument.Save(stream, FormatType.Docx);
destinationDocument.Close();
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Hello
World.docx"), FormatType.Docx))
{
//Opens the destination document

```

```

WordDocument destinationDocument = new
WordDocument(assembly.GetManifestResourceStream("XamarinFormsAppl.Assets.Let
ter Formatting.docx"), FormatType.Docx);
//Imports the contents of source document at the end of destination document
destinationDocument.ImportContent(document,
ImportOptions.UseDestinationStyles);
MemoryStream stream = new MemoryStream();
destinationDocument.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWordDoc.doc
x", "application/msword", stream);
//Closes the documents
document.Close();
destinationDocument.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
}

```

In the resultant document, the imported contents start from a new page followed by existing contents in a destination document. This is the default behavior.

When your requirement is to append the contents from the same page instead of starting from a new page, you need to set the break code of first section of Source document as NoBreak. The following code example illustrates the importing contents from the same page.

C#

```

//Opens the source document
WordDocument sourceDocument = new WordDocument(sourceFileName);
//Opens the destination document
WordDocument destinationDocument = new WordDocument(targetFileName);
//Sets the break-code of First section of source document as NoBreak to
avoid imported from a new page
sourceDocument.Sections[0].BreakCode = SectionBreakCode.NoBreak;
//Imports the contents of source document at the end of destination document
destinationDocument.ImportContent(sourceDocument,
ImportOptions.UseDestinationStyles);
//Saves the destination document
destinationDocument.Save(outputFileName, FormatType.Docx);
//Closes the document instances
sourceDocument.Close();
destinationDocument.Close();

```

VB.NET

```

'Opens the source document
Dim sourceDocument As New WordDocument(sourceFileName)
'Opens the destination document
Dim destinationDocument As New WordDocument(targetFileName)
'Sets the break-code of first section of source document as NoBreak to avoid
imported from a new page
sourceDocument.Sections(0).BreakCode = SectionBreakCode.NoBreak
'Imports the contents of source document at the end of destination document

```

```

destinationDocument.ImportContent(sourceDocument,
ImportOptions.UseDestinationStyles)
'Saves the destination document
destinationDocument.Save(outputFileName, FormatType.Docx)
'Closes the document instances
sourceDocument.Close()
destinationDocument.Close()

```

UWP

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Tes
t.docx"), FormatType.Docx))
{
    // Opens the destination document
    WordDocument destinationDocument = new
    WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Let
ter Formatting.docx"), FormatType.Docx);
    // Sets the break-code of First section of source document as NoBreak to
    avoid imported from a new page
    document.Sections[0].BreakCode = SectionBreakCode.NoBreak;
    // Imports the contents of source document at the end of destination document
    destinationDocument.ImportContent(document,
    ImportOptions.UseDestinationStyles);
    MemoryStream stream = new MemoryStream();
    // Saves the Word file to MemoryStream
    await destinationDocument.SaveAsync(stream, FormatType.Docx);
    // Saves the stream as Word file in local machine
    Save(stream, "Result.docx");
    // Please refer the below link to save Word document in UWP platform
    // https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
    document.Close();
    destinationDocument.Close();
}

```

ASP.NET CORE

```

FileStream sourceStreamPath = new FileStream(sourceFileName, FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
FileStream destinationStreamPath = new FileStream(destinationFileName,
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
// Opens an source document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(sourceStreamPath,
FormatType.Automatic))
{
    // Opens the destination document
    WordDocument destinationDocument = new WordDocument(destinationStreamPath,
    FormatType.Docx);
    // Sets the break-code of First section of source document as NoBreak to
    avoid imported from a new page
    document.Sections[0].BreakCode = SectionBreakCode.NoBreak;
    // Imports the contents of source document at the end of destination document

```

```

destinationDocument.ImportContent(document,
ImportOptions.UseDestinationStyles);
MemoryStream stream = new MemoryStream();
//Saves and closes the destination document to MemoryStream
destinationDocument.Save(stream, FormatType.Docx);
destinationDocument.Close();
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.HelloWorld.docx"), FormatType.Docx))
{
//Opens the destination document
WordDocument destinationDocument = new
WordDocument(assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.LetterFormatting.docx"), FormatType.Docx);
//Sets the break-code of First section of source document as NoBreak to
avoid imported from a new page
document.Sections[0].BreakCode = SectionBreakCode.NoBreak;
//Imports the contents of source document at the end of destination document
destinationDocument.ImportContent(document,
ImportOptions.UseDestinationStyles);
MemoryStream stream = new MemoryStream();
destinationDocument.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWordDoc.docx", "application/msword", stream);
//Closes the documents
document.Close();
destinationDocument.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
}

```

Maintain Imported List style information

The following code example shows how to maintain information about imported list styles in a Word document while cloning and merging multiple Word documents.

C#

```

//Opens the source document
WordDocument sourceDocument = new WordDocument(sourceFileName);
//Opens the destination document
WordDocument destinationDocument = new WordDocument(targetFileName);

```

```

//Sets true value to maintain imported list style cache to destination document
destinationDocument.Settings.MaintainImportedListCache = true;
//Processes the body contents for each section in the Word document
foreach (WSection section in sourceDocument.Sections)
{
    //Accesses the body of section where all the contents in document are apart
    foreach (TextBodyItem bodyItem in section.Body.ChildEntities)
    {
        destinationDocument.LastSection.Body.ChildEntities.Add(bodyItem.Clone());
    }
}
//Closes the source document
sourceDocument.Close();
//Sets false value to exclude imported list style cache to destination document
destinationDocument.Settings.MaintainImportedListCache = false;
//Saves the destination document
destinationDocument.Save(outputFileName, FormatType.Docx);
//Closes the destination document
destinationDocument.Close();

```

VB.NET

```

'Opens the source document
Dim sourceDocument As New WordDocument(sourceFileName)
'Opens the destination document
Dim destinationDocument As New WordDocument(targetFileName)
'Sets true value to maintain imported list style cache to destination document
destinationDocument.Settings.MaintainImportedListCache = True
'Processes the body contents for each section in the Word document
For Each section As WSection In sourceDocument.Sections
    'Accesses the body of section where all the contents in document are apart
    For Each bodyItem As TextBodyItem In section.Body.ChildEntities
        destinationDocument.LastSection.Body.ChildEntities.Add(bodyItem.Clone())
    Next
Next
'Closes the source document
sourceDocument.Close()
'Sets false value to exclude imported list style cache to destination document
destinationDocument.Settings.MaintainImportedListCache = False
'Saves the destination document
destinationDocument.Save(outputFileName, FormatType.Docx)
'Closes the destination document
destinationDocument.Close()

```

UWP

```

//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument sourceDocument = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Source.docx"),
FormatType.Docx);

```

```

WordDocument destinationDocument = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Destination.d
ocx"), FormatType.Docx);
//Sets true value to maintain imported list style cache to destination
document
destinationDocument.Settings.MaintainImportedListCache = true;
//Processes the body contents for each section in the Word document
foreach (WSection section in sourceDocument.Sections)
{
//Accesses the body of section where all the contents in document are apart
foreach (TextBodyItem bodyItem in section.Body.ChildEntities)
{
destinationDocument.LastSection.Body.ChildEntities.Add(bodyItem.Clone());
}
}
//Closes the source document
sourceDocument.Close();
//Sets false value to exclude imported list style cache to destination
document
destinationDocument.Settings.MaintainImportedListCache = false;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await destinationDocument.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
destinationDocument.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Opens the source document
FileStream SourceFileStream = new FileStream("Source.docx", FileMode.Open);
WordDocument sourceDocument = new WordDocument(SourceFileStream,
FormatType.Docx);
//Opens the destination document
FileStream DestinationFileStream = new FileStream("Destination.docx",
FileMode.Open);
WordDocument destinationDocument = new WordDocument(DestinationFileStream,
FormatType.Docx);
//Sets true value to maintain imported list style cache to destination
document
destinationDocument.Settings.MaintainImportedListCache = true;
//Processes the body contents for each section in the Word document
foreach (WSection section in sourceDocument.Sections)
{
//Accesses the body of section where all the contents in document are apart
foreach (TextBodyItem bodyItem in section.Body.ChildEntities)
{
destinationDocument.LastSection.Body.ChildEntities.Add(bodyItem.Clone());
}
}
//Closes the source document
sourceDocument.Close();

```

```
//Sets false value to exclude imported list style cache to destination
document
destinationDocument.Settings.MaintainImportedListCache = false;
//Saves the destination document
MemoryStream outputStream = new MemoryStream();
destinationDocument.Save(outputStream, FormatType.Docx);
//Closes the destination document
destinationDocument.Close();
outputStream.Position = 0;
//Download Word document in the browser
return File(outputStream, "application/msword", "Result.docx");
```

XAMARIN

```
//Opens the source document
Stream sourceStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Source.docx");
WordDocument sourceDocument = new WordDocument(sourceStream,
FormatType.Docx);
//Opens the destination document
Stream destinationStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Destination.docx");
WordDocument destinationDocument = new WordDocument(destinationStream,
FormatType.Docx);
//Sets true value to maintain imported list style cache to destination
document
destinationDocument.Settings.MaintainImportedListCache = true;
//Processes the body contents for each section in the Word document
foreach (WSection section in sourceDocument.Sections)
{
//Accesses the body of section where all the contents in document are apart
foreach (TextBodyItem bodyItem in section.Body.ChildEntities)
{
destinationDocument.LastSection.Body.ChildEntities.Add(bodyItem.Clone());
}
}
//Closes the source document
sourceDocument.Close();
//Sets false value to exclude imported list style cache to destination
document
destinationDocument.Settings.MaintainImportedListCache = false;
//Saves the destination document
MemoryStream outputStream = new MemoryStream();
destinationDocument.Save(outputStream, FormatType.Docx);
//Closes the destination document
destinationDocument.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Printing a Word document

You can print a Word document by utilizing DocIO's capability to convert the document into images and .NET framework's [PrintDocument](#) class

Initially you have to render the pages as images as shown below

C#

```
//Opens the Word document
WordDocument document = new WordDocument((string) this.textBox.Tag);
//Renders the Word document as image
Image[] images = document.RenderAsImages(ImageType.Metafile);
//Closes the Word Document
document.Close();
```

VB.NET

```
'Opens the Word document
Dim document As New WordDocument(DirectCast(Me.textBox.Tag, String))
'Renders the Word document as image
Dim images As Image() = document.RenderAsImages(ImageType.Metafile)
'Closes the Word Document
document.Close()
```

UWP

```
//DocIO supports Word to Image conversion in Windows forms, WPF, ASP.NET and ASP.NET MVC platforms only.
```

ASP.NET CORE

```
//DocIO supports Word to Image conversion in Windows forms, WPF, ASP.NET and ASP.NET MVC platforms only.
```

XAMARIN

```
//DocIO supports Word to Image conversion in Windows forms, WPF, ASP.NET and ASP.NET MVC platforms only.
```

You can specify the printer settings and page settings through the [PrintDocument](#) class. The [PrintDocument.PrintPage](#) event should be handled to layout the document for printing.

The following code example demonstrates how to print the Word document pages that have been rendered as an image:

C#

```
int endPageIndex = images.Length;
//Creates new PrintDialog instance
System.Windows.Forms.PrintDialog printDialog = new
System.Windows.Forms.PrintDialog();
//Sets new PrintDocument instance to print dialog
printDialog.Document = new PrintDocument();
//Enables the print current page option
printDialog.AllowCurrentPage = true;
```



```

//Enables the print selected pages option
printDialog.AllowSomePages = true;
//Sets the start and end page index
printDialog.PrinterSettings.FromPage = 1;
printDialog.PrinterSettings.ToPage = images.Length;
//Opens the print dialog box
if (printDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
{
    //Checks whether the selected page range is valid
    if (printDialog.PrinterSettings.FromPage > 0 &&
        printDialog.PrinterSettings.ToPage <= images.Length)
    {
        //Updates the start page of the document to print
        startPageIndex = printDialog.PrinterSettings.FromPage - 1;
        //Updates the end page of the document to print
        endPageIndex = printDialog.PrinterSettings.ToPage;
        //Hooks the PrintPage event to handle the drawing pages for printing
        printDialog.Document.PrintPage += new
        PrintPageEventHandler(PrintPageMethod);
        //Print the document
        printDialog.Document.Print();
    }
}

```

VB.NET

```

Dim endPageIndex As Integer = images.Length
'Creates new PrintDialog instance
Dim printDialog As New System.Windows.Forms.PrintDialog()
'Sets new PrintDocument instance to print dialog
printDialog.Document = New PrintDocument()
'Enables the print current page option
printDialog.AllowCurrentPage = True
'Enables the print selected pages option
printDialog.AllowSomePages = True
'Sets the start and end page index
printDialog.PrinterSettings.FromPage = 1
printDialog.PrinterSettings.ToPage = images.Length
'Opens the print dialog box
If printDialog.ShowDialog() = System.Windows.Forms.DialogResult.OK Then
    'Checks whether the selected page range is valid or not
    If printDialog.PrinterSettings.FromPage > 0 AndAlso
    printDialog.PrinterSettings.ToPage <= images.Length Then
        'Updates the start page of the document to print
        startPageIndex = printDialog.PrinterSettings.FromPage - 1
        'Updates the end page of the document to print
        endPageIndex = printDialog.PrinterSettings.ToPage
        'Hooks the PrintPage event to handle the drawing pages for printing
        printDialog.Document.PrintPage += New PrintPageEventHandler(PrintPageMethod)
        'Prints the document
        printDialog.Document.Print()
    End If
End If

```

UWP

```
//DocIO supports Word to Image conversion in Windows forms, WPF, ASP.NET and ASP.NET MVC platforms only.
```

ASP.NET CORE

```
//DocIO supports Word to Image conversion in Windows forms, WPF, ASP.NET and ASP.NET MVC platforms only.
```

XAMARIN

```
//DocIO supports Word to Image conversion in Windows forms, WPF, ASP.NET and ASP.NET MVC platforms only.
```

The following code example provides supporting methods for the above code.

C#

```
private void PrintPageMethod(object sender, PrintPageEventArgs e)
{
    //Gets the print start page width
    int currentPageWidth = images[startPageIndex].Width;
    //Gets the print start page height
    int currentPageHeight = images[startPageIndex].Height;
    //Gets the visible bounds width for print
    int visibleClipBoundsWidth = (int)e.Graphics.VisibleClipBounds.Width;
    //Gets the visible bounds height for print
    int visibleClipBoundsHeight = (int)e.Graphics.VisibleClipBounds.Height;
    //Checks whether the page layout is landscape or portrait
    if (currentPageWidth > currentPageHeight)
    {
        //Translates the position
        e.Graphics.TranslateTransform(0, visibleClipBoundsHeight);
        //Rotates the object at 270 degrees
        e.Graphics.RotateTransform(270.0f);
        //Draws the current page image
        e.Graphics.DrawImage(images[startPageIndex], new System.Drawing.Rectangle(0, 0, currentPageWidth, currentPageHeight));
    }
    else
    {
        //Draws the current page image
        e.Graphics.DrawImage(images[startPageIndex], new System.Drawing.Rectangle(0, 0, visibleClipBoundsWidth, visibleClipBoundsHeight));
    }
    //Disposes the current page image after drawing
    images[startPageIndex].Dispose();
    //Increments the start page index
    startPageIndex++;
    //Updates whether the document contains some more pages to print
    if (startPageIndex < endPageIndex)
        e.HasMorePages = true;
    else
        startPageIndex = 0;
}
```

VB.NET

```

Private Sub PrintPageMethod(sender As Object, e As PrintPageEventArgs)
    'Gets the print start page width
    Dim currentPageWidth As Integer = images(startPageIndex).Width
    'Gets the print start page height
    Dim currentPageHeight As Integer = images(startPageIndex).Height
    'Gets the visible bounds width for print
    Dim visibleClipBoundsWidth As Integer =
    CInt(e.Graphics.VisibleClipBounds.Width)
    'Gets the visible bounds height for print
    Dim visibleClipBoundsHeight As Integer =
    CInt(e.Graphics.VisibleClipBounds.Height)
    'Checks whether the page layout is landscape or portrait
    If currentPageWidth > currentPageHeight Then
        'Translates the position
        e.Graphics.TranslateTransform(0, visibleClipBoundsHeight)
        'Rotates the object at 270 degrees
        e.Graphics.RotateTransform(270.0F)
        'Draws the current page image
        e.Graphics.DrawImage(images(startPageIndex), New System.Drawing.Rectangle(0,
        0, currentPageWidth, currentPageHeight))
    Else
        'Draws the current page image
        e.Graphics.DrawImage(images(startPageIndex), New System.Drawing.Rectangle(0,
        0, visibleClipBoundsWidth, visibleClipBoundsHeight))
    End If
    'Disposes the current page image after drawing
    images(startPageIndex).Dispose()
    'Increments the start page index
    startPageIndex += 1
    'Updates whether the document contains some more pages to print
    If startPageIndex < endPageIndex Then
        e.HasMorePages = True
    Else
        startPageIndex = 0
    End If
End Sub

```

UWP

```

//DocIO supports Word to Image conversion in Windows forms, WPF, ASP.NET and
ASP.NET MVC platforms only.

```

ASP.NET CORE

```

//DocIO supports Word to Image conversion in Windows forms, WPF, ASP.NET and
ASP.NET MVC platforms only.

```

XAMARIN

```

//DocIO supports Word to Image conversion in Windows forms, WPF, ASP.NET and
ASP.NET MVC platforms only.

```

You can download the complete working samples of the code from [here](#).

Working with Styles

A style is a predefined set of table, numbering, paragraph, and character properties that can be applied to regions within a document. DocIO provides the following functionalities related with styles.

- Access and modify the existing styles in the word document
- Create new paragraph style.
- Apply built-in styles.

Access Styles

Paragraph and character styles present in the existing document are accessible through the **Styles** property of **WordDocument** class.

This following code example demonstrates how a style can be accessed and style properties like text color and first line indent can be updated.

C#

```
//Opens an input Word template
WordDocument document = new WordDocument(inputFileName);
//Accesses the styles collection that contains paragraph and character
styles in Word document
IStyleCollection styleCollection = document.Styles;
//Finds the style with the name "Heading 1"
WParagraphStyle heading1ParagraphStyle = styleCollection.FindByName("Heading
1") as WParagraphStyle;
//Changes the text color of style "Heading 1" as DarkBlue
heading1ParagraphStyle.CharacterFormat.TextColor = Color.DarkBlue;
//Changes the first line indent of Paragraph as 36 points
heading1ParagraphStyle.ParagraphFormat.FirstLineIndent = 36;
//Saves and closes the document instance
document.Save(outputFileName, FormatType.Docx);
document.Close();
```

VB.NET

```
'Opens an input Word template
Dim document As New WordDocument(inputFileName)
'Accesses the styles collection that contains paragraph and character styles
in Word document
Dim styleCollection As IStyleCollection = document.Styles
'Finds the style with the name "Heading 1"
Dim heading1ParagraphStyle As WParagraphStyle =
TryCast(styleCollection.FindByName("Heading 1"), WParagraphStyle)
'Changes the text color of style "Heading 1" as DarkBlue
heading1ParagraphStyle.CharacterFormat.TextColor = Color.DarkBlue
'Changes the first line indent of paragraph as 36 points
heading1ParagraphStyle.ParagraphFormat.FirstLineIndent = 36
'Saves and closes the document instance
document.Save(outputFileName, FormatType.Docx)
document.Close()
```

UWP

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Tes
t.docx"), FormatType.Docx))
{
//Accesses the styles collection that contains paragraph and character
styles in Word document
IStyleCollection styleCollection = document.Styles;
//Finds the style with the name "Heading 1"
WParagraphStyle heading1ParagraphStyle = styleCollection.FindByName("Heading
1") as WParagraphStyle;
//Changes the text color of style "Heading 1" as DarkBlue
heading1ParagraphStyle.CharacterFormat.TextColor =
Syncfusion.DocIO.DLS.Color.DarkBlue;
//Changes the first line indent of Paragraph as 36 points
heading1ParagraphStyle.ParagraphFormat.FirstLineIndent = 36;
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
document.Close();
}
```

ASP.NET CORE

```
FileStream sourceStreamPath = new FileStream(sourceFileName, FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
//Opens an source document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(sourceStreamPath,
FormatType.Automatic))
{
//Accesses the styles collection that contains paragraph and character
styles in Word document
IStyleCollection styleCollection = document.Styles;
//Finds the style with the name "Heading 1"
WParagraphStyle heading1ParagraphStyle = styleCollection.FindByName("Heading
1") as WParagraphStyle;
//Changes the text color of style "Heading 1" as DarkBlue
heading1ParagraphStyle.CharacterFormat.TextColor =
Syncfusion.Drawing.Color.DarkBlue;
//Changes the first line indent of Paragraph as 36 points
heading1ParagraphStyle.ParagraphFormat.FirstLineIndent = 36;
MemoryStream stream = new MemoryStream();
//Saves and closes the destination document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}
```

XAMARIN

```

///"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Hello World.docx"), FormatType.Docx))
{
//Accesses the styles collection that contains paragraph and character styles in Word document
IStyleCollection styleCollection = document.Styles;
//Finds the style with the name "Heading 1"
WParagraphStyle heading1ParagraphStyle = styleCollection.FindByName("Heading 1") as WParagraphStyle;
//Changes the text color of style "Heading 1" as DarkBlue
heading1ParagraphStyle.CharacterFormat.TextColor = Syncfusion.Drawing.Color.DarkBlue;
//Changes the first line indent of Paragraph as 36 points
heading1ParagraphStyle.ParagraphFormat.FirstLineIndent = 36;
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWordDoc.docx", "application/msword", stream);
//Closes the document
document.Close();
//Please download the helper files from the below link to save the stream as file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
}

```

Creating a new Paragraph Style

You can create a new paragraph style by using `WordDocument.AddParagraphStyle` method and apply it by using `ApplyStyle` method of `WParagraph` class.

C#

```

//Opens an input Word template
WordDocument document = new WordDocument();
//This method adds a section and a paragraph in the document
document.EnsureMinimal();
//Adds a new paragraph style named "MyStyle"
IWParagraphStyle myStyle = document.AddParagraphStyle("MyStyle");
//Sets the formatting of the style
myStyle.CharacterFormat.FontSize = 16f;
myStyle.CharacterFormat.TextColor = Color.DarkBlue;
myStyle.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
//Appends the contents into the paragraph
document.LastParagraph.AppendText("AdventureWorks Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company.");
//Applies the style to paragraph
document.LastParagraph.ApplyStyle("MyStyle");

```

```
document.Save(outputFileName, FormatType.Docx);
document.Close();
```

VB.NET

```
'Opens an input Word template
Dim document As New WordDocument()
'This method adds a section and a paragraph in the document
document.EnsureMinimal()
'Adds a new paragraph style named "MyStyle"
Dim myStyle As IWPParagraphStyle = document.AddParagraphStyle("MyStyle")
'Sets the formatting of the style
myStyle.CharacterFormat.FontSize = 16.0F
myStyle.CharacterFormat.TextColor = Color.DarkBlue
myStyle.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right
'Appends the content into the paragraph
document.LastParagraph.AppendText("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.")
'Applies the style to paragraph
document.LastParagraph.ApplyStyle("MyStyle")
document.Save(outputFileName, FormatType.Docx)
document.Close()
```

UWP

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Tes
t.docx"), FormatType.Docx))
{
IWPParagraphStyle myStyle = document.AddParagraphStyle("MyStyle");
//Sets the formatting of the style
myStyle.CharacterFormat.FontSize = 16f;
myStyle.CharacterFormat.TextColor = Syncfusion.DocIO.DLS.Color.DarkBlue;
myStyle.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Right;
//Appends the contents into the paragraph
document.LastParagraph.AppendText("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.");
//Applies the style to paragraph
document.LastParagraph.ApplyStyle("MyStyle");
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
document.Close();
}
```

ASP.NET CORE

```

FileStream sourceStreamPath = new FileStream(sourceFileName, FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
//Opens an source document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(sourceStreamPath,
FormatType.Automatic))
{
IWParagraphStyle myStyle = document.AddParagraphStyle("MyStyle");
//Sets the formatting of the style
myStyle.CharacterFormat.FontSize = 16f;
myStyle.CharacterFormat.TextColor = Syncfusion.Drawing.Color.DarkBlue;
myStyle.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
//Appends the contents into the paragraph
document.LastParagraph.AppendText("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.");
//Applies the style to paragraph
document.LastParagraph.ApplyStyle("MyStyle");
MemoryStream stream = new MemoryStream();
//Saves and closes the destination document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Hello
World.docx"), FormatType.Docx))
{
IWParagraphStyle myStyle = document.AddParagraphStyle("MyStyle");
//Sets the formatting of the style
myStyle.CharacterFormat.FontSize = 16f;
myStyle.CharacterFormat.TextColor = Syncfusion.Drawing.Color.DarkBlue;
myStyle.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
//Appends the contents into the paragraph
document.LastParagraph.AppendText("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.");
//Applies the style to paragraph
document.LastParagraph.ApplyStyle("MyStyle");
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWordDoc.doc
x", "application/msword", stream);
//Closes the document
document.Close();
}

```



```
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
}
```

Applying built-in styles

DocIO provides a set of predefined styles. You can apply those predefined styles as shown in the following code example.

C#

```
//Opens an input Word template
WordDocument document = new WordDocument();
//This method adds a section and a paragraph in the document
document.EnsureMinimal();
IWParagraph paragraph = document.LastParagraph;
//Appends the content into the paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Applies the style to paragraph
paragraph.ApplyStyle(BuiltinStyle.Emphasis);
document.Save(outputFileName, FormatType.Docx);
document.Close();
```

VB.NET

```
'Opens an input Word template
Dim document As New WordDocument()
'This method adds a section and a paragraph in the document
document.EnsureMinimal()
Dim paragraph As IWParagraph = document.LastParagraph
'Appends the content into the paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.")
'Applies the style to paragraph
paragraph.ApplyStyle(BuiltinStyle.Emphasis)
document.Save(outputFileName, FormatType.Docx)
document.Close()
```

UWP

```
/"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Tes
t.docx"), FormatType.Docx))
{
//Applies the style to paragraph
document.LastParagraph.ApplyStyle(BuiltinStyle.Emphasis);
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
```

```
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
document.Close();
}
```

ASP.NET CORE

```
FileStream sourceStreamPath = new FileStream(sourceFileName, FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
//Opens an source document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(sourceStreamPath,
FormatType.Automatic))
{
//Applies the style to paragraph
document.LastParagraph.ApplyStyle(BuiltinStyle.Emphasis);
MemoryStream stream = new MemoryStream();
//Saves and closes the destination document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}
```

XAMARIN

```
///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("XamarinFormsAppl.Assets.Hello World.docx"), FormatType.Docx))
{
//Applies the style to paragraph
document.LastParagraph.ApplyStyle(BuiltinStyle.Emphasis);
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWordDoc.docx", "application/msword", stream);
//Closes the document
document.Close();
//Please download the helper files from the below link to save the stream as file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
}
```

Working with Word document properties

Document properties, also known as metadata, are details about a file that describe or identify it. You can also define the additional custom document properties for the documents by using DocIO

Document properties that are classified as two categories.

- Built-in document properties - includes details such as title, author name, subject, and keywords that identify the document's topic or contents.
- Custom document properties - defines the user-defined document properties.

Built-in document properties

The Built-in document properties of a word document is represented by `BuiltinDocumentProperties` property of `WordDocument` class. The following code example illustrates how to access and modify the Built-in document properties of the document.

C#

```
//Opens an existing Word document
WordDocument document = new WordDocument(inputFileName);
//Accesses the built-in document properties
Console.WriteLine("Title - {0}", document.BuiltinDocumentProperties.Title);
Console.WriteLine("Author - {0}",
document.BuiltinDocumentProperties.Author);
//Modifies or sets the category and company Built-in document properties
document.BuiltinDocumentProperties.Category = "Sales reports";
document.BuiltinDocumentProperties.Company = "Northwind traders";
document.Save(outputFileName, FormatType.Docx);
document.Close();
```

VB.NET

```
'Opens an existing Word document
Dim document As New WordDocument(inputFileName)
'Accesses the built-in document properties
Console.WriteLine("Title - {0}", document.BuiltinDocumentProperties.Title)
Console.WriteLine("Author - {0}", document.BuiltinDocumentProperties.Author)
'Modifies or sets the category and company Built-in document properties
document.BuiltinDocumentProperties.Category = "Sales reports"
document.BuiltinDocumentProperties.Company = "Northwind traders"
'Saves and closes the document
document.Save(outputFileName, FormatType.Docx)
document.Close()
```

UWP

```
///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Tes
t.docx"), FormatType.Docx))
{
    ///Accesses the built-in document properties
    Console.WriteLine("Title - {0}", document.BuiltinDocumentProperties.Title);
```

```

Console.WriteLine("Author - {0}",
document.BuiltinDocumentProperties.Author);
//Modifies or sets the category and company Built-in document properties
document.BuiltinDocumentProperties.Category = "Sales reports";
document.BuiltinDocumentProperties.Company = "Northwind traders";
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
document.Close();
}

```

ASP.NET CORE

```

FileStream sourceStreamPath = new FileStream(sourceFileName, FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
//Opens an source document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(sourceStreamPath,
FormatTypeAutomatic))
{
//Accesses the built-in document properties
Console.WriteLine("Title - {0}", document.BuiltinDocumentProperties.Title);
Console.WriteLine("Author - {0}",
document.BuiltinDocumentProperties.Author);
//Modifies or sets the category and company Built-in document properties
document.BuiltinDocumentProperties.Category = "Sales reports";
document.BuiltinDocumentProperties.Company = "Northwind traders";
MemoryStream stream = new MemoryStream();
//Saves and closes the destination document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

//App is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Hello World.docx"), FormatType.Docx))
{
//Accesses the built-in document properties
Console.WriteLine("Title - {0}", document.BuiltinDocumentProperties.Title);
Console.WriteLine("Author - {0}",
document.BuiltinDocumentProperties.Author);
//Modifies or sets the category and company Built-in document properties
document.BuiltinDocumentProperties.Category = "Sales reports";
document.BuiltinDocumentProperties.Company = "Northwind traders";
}

```

```

MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWordDoc.doc
x", "application/msword", stream);
//Closes the document
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
}

```

Adding Custom Document properties

You add a new custom document properties through **Add** method of **CustomProperties** class. The following code example illustrates how to add a new custom document properties.

C#

```

//Opens an input word template
WordDocument document = new WordDocument(inputFileName);
//Adds the custom document properties of various data types
document.CustomDocumentProperties.Add("PropertyA", "Value of A");
document.CustomDocumentProperties.Add("PropertyB", 12.5);
document.CustomDocumentProperties.Add("PropertyC", true);
document.CustomDocumentProperties.Add("PropertyD", new DateTime(2015, 7, 20));
//Saves and closes the document
document.Save(outputFileName, FormatType.Docx);
document.Close();

```

VB.NET

```

'Opens an existing document from file system through constructor of
WordDocument class
Dim document As New WordDocument(inputFileName)
'Adds the custom document properties of various data types
document.CustomDocumentProperties.Add("PropertyA", "Value of A")
document.CustomDocumentProperties.Add("PropertyB", 12.5)
document.CustomDocumentProperties.Add("PropertyC", True)
document.CustomDocumentProperties.Add("PropertyD", New DateTime(2015, 7,
20))
'Saves and closes the document
document.Save(outputFileName, FormatType.Docx)
document.Close()

```

UWP

```

//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Tes
t.docx"), FormatType.Docx))
{
//Adds the custom document properties of various data types
document.CustomDocumentProperties.Add("PropertyA", "Value of A");
}

```

```

document.CustomDocumentProperties.Add("PropertyB", 12.5);
document.CustomDocumentProperties.Add("PropertyC", true);
document.CustomDocumentProperties.Add("PropertyD", new DateTime(2015, 7, 20));
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
document.Close();
}

```

ASP.NET CORE

```

FileStream sourceStreamPath = new FileStream(sourceFileName, FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
//Opens an source document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(sourceStreamPath,
FormatType.Automatic))
{
//Adds the custom document properties of various data types
document.CustomDocumentProperties.Add("PropertyA", "Value of A");
document.CustomDocumentProperties.Add("PropertyB", 12.5);
document.CustomDocumentProperties.Add("PropertyC", true);
document.CustomDocumentProperties.Add("PropertyD", new DateTime(2015, 7, 20));
MemoryStream stream = new MemoryStream();
//Saves and closes the destination document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("XamarinFormsAppl.Assets.Hello World.docx"), FormatType.Docx))
{
//Adds the custom document properties of various data types
document.CustomDocumentProperties.Add("PropertyA", "Value of A");
document.CustomDocumentProperties.Add("PropertyB", 12.5);
document.CustomDocumentProperties.Add("PropertyC", true);
document.CustomDocumentProperties.Add("PropertyD", new DateTime(2015, 7, 20));
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWordDoc.docx", "application/msword", stream);
//Closes the document
}

```

```
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
}
```

Accessing & Modifying Custom Document Properties

You can access and modify an existing document property as shown in the following code example.

C#

```
WordDocument document = new WordDocument(inputFileName);
//Accesses an existing custom document property
DocumentProperty property = document.CustomDocumentProperties["PropertyA"];
//Modifies the value of DocumentProperty instance
property.Value = "Hello world";
document.Save(outputFileName, FormatType.Docx);
document.Close();
```

VB.NET

```
Dim document As New WordDocument(inputFileName)
'Accesses an existing custom document property
Dim [property] As DocumentProperty =
document.CustomDocumentProperties("PropertyA")
'Modifies the value of DocumentProperty instance
[property].Value = "Hello world"
document.Save(outputFileName, FormatType.Docx)
document.Close()
```

UWP

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Tes
t.docx"), FormatType.Docx))
{
//Accesses an existing custom document property
DocumentProperty property = document.CustomDocumentProperties["PropertyA"];
//Modifies the value of DocumentProperty instance
property.Value = "Hello world";
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
document.Close();
}
```

ASP.NET CORE

```

FileStream sourceStreamPath = new FileStream(sourceFileName, FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
//Opens an source document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(sourceStreamPath,
FormatType.Automatic))
{
    //Accesses an existing custom document property
    DocumentProperty property = document.CustomDocumentProperties["PropertyA"];
    //Modifies the value of DocumentProperty instance
    property.Value = "Hello world";
    MemoryStream stream = new MemoryStream();
    //Saves and closes the destination document to MemoryStream
    document.Save(stream, FormatType.Docx);
    document.Close();
    stream.Position = 0;
    //Download Word document in the browser
    return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Hello World.docx"), FormatType.Docx))
{
    //Accesses an existing custom document property
    DocumentProperty property = document.CustomDocumentProperties["PropertyA"];
    //Modifies the value of DocumentProperty instance
    property.Value = "Hello world";
    MemoryStream stream = new MemoryStream();
    document.Save(stream, FormatType.Docx);
    //Save the stream as a file in the device and invoke it for viewing
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWordDoc.docx", "application/msword", stream);
    //Closes the document
    document.Close();
    //Please download the helper files from the below link to save the stream as file and open the file for viewing in Xamarin platform
    //https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
}

```

Setting the Background for a Word document

Essential DocIO allows to apply background such as color, gradient and picture to the Word document. A background of a Word document is represented by **Background** property of 'WordDocument' class.

The following code illustrates how to apply gradient as background to the document.

C#

```

//Creates a new Word document

```



```

WordDocument document = new WordDocument();
//Adds new section to the document
WSection section = document.AddSection() as WSection;
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph() as WParagraph;
//Appends text to the paragraph
paragraph.AppendText("Sample for applying document background");
//Sets the background type as gradient
document.Background.Type = BackgroundType.Gradient;
//Sets color for gradient
document.Background.Gradient.Color1 = Color.LightGray;
document.Background.Gradient.Color2 = Color.LightGreen;
//Sets the shading style
document.Background.Gradient.ShadingStyle = GradientShadingStyle.DiagonalUp;
document.Background.Gradient.ShadingVariant =
GradientShadingVariant.ShadingDown;
//Saves the document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As WSection = TryCast(document.AddSection(), WSection)
'Adds new paragraph to the section
Dim paragraph As IWParagraph = TryCast(section.AddParagraph(), WParagraph)
'Appends text to the paragraph
paragraph.AppendText("Sample for applying document background")
'Sets the background type as gradient
document.Background.Type = BackgroundType.Gradient
'Sets color for gradient
document.Background.Gradient.Color1 = Color.LightGray
document.Background.Gradient.Color2 = Color.LightGreen
'Sets the shading style
document.Background.Gradient.ShadingStyle = GradientShadingStyle.DiagonalUp
document.Background.Gradient.ShadingVariant =
GradientShadingVariant.ShadingDown
'Saves the document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Tes
t.docx"), FormatType.Docx))
{
document.Background.Type = BackgroundType.Picture;
//Sets color for gradient
document.Background.Gradient.Color1 = Syncfusion.DocIO.DLS.Color.LightGray;

```

```

document.Background.Gradient.Color2 = Syncfusion.DocIO.DLS.Color.LightGreen;
//Sets the shading style
document.Background.Gradient.ShadingStyle = GradientShadingStyle.DiagonalUp;
document.Background.Gradient.ShadingVariant =
GradientShadingVariant.ShadingDown;
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
document.Close();
}

```

ASP.NET CORE

```

FileStream sourceStreamPath = new FileStream(sourceFileName, FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
//Opens an source document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(sourceStreamPath,
FormatTypeAutomatic))
{
//Sets the background type as picture
document.Background.Type = BackgroundType.Picture;
//Sets color for gradient
document.Background.Gradient.Color1 = Syncfusion.Drawing.Color.LightGray;
document.Background.Gradient.Color2 = Syncfusion.Drawing.Color.LightGreen;
//Sets the shading style
document.Background.Gradient.ShadingStyle = GradientShadingStyle.DiagonalUp;
document.Background.Gradient.ShadingVariant =
GradientShadingVariant.ShadingDown;
MemoryStream stream = new MemoryStream();
//Saves and closes the destination document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Hello
World.docx"), FormatType.Docx))
{
document.Background.Type = BackgroundType.Picture;
//Sets color for gradient
document.Background.Gradient.Color1 = Syncfusion.Drawing.Color.LightGray;
document.Background.Gradient.Color2 = Syncfusion.Drawing.Color.LightGreen;
//Sets the shading style

```

```

document.Background.Gradient.ShadingStyle = GradientShadingStyle.DiagonalUp;
document.Background.Gradient.ShadingVariant =
GradientShadingVariant.ShadingDown;
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWordDoc.doc
x", "application/msword", stream);
//Closes the document
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
}

```

The following code illustrates how to apply image as background for the document.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
WSection section = document.AddSection() as WSection;
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph() as WParagraph;
//Appends text to the paragraph
paragraph.AppendText("Sample for applying document background");
//Sets the background type as picture
document.Background.Type = BackgroundType.Picture;
document.Background.Picture = Image.FromFile("Image.png");
//Saves the document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to document
Dim section As WSection = TryCast(document.AddSection(), WSection)
'Adds new paragraph to the section
Dim paragraph As IWParagraph = TryCast(section.AddParagraph(), WParagraph)
'Appends text to the paragraph
paragraph.AppendText("Sample for applying document background")
'Sets the background type as picture
document.Background.Type = BackgroundType.Picture
document.Background.Picture = Image.FromFile("Image.png")
'Saves the document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("CreateWordSample.Assets.Tes
t.docx"), FormatType.Docx))
{
document.Background.Type = BackgroundType.Picture;
//Opens the existing image
Stream imageStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Picture.png");
MemoryStream memoryStream = new MemoryStream();
imageStream.CopyTo(memoryStream);
document.Background.Picture = memoryStream.ToArray();
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
document.Close();
}
```

ASP.NET CORE

```
FileStream sourceStreamPath = new FileStream(sourceFileName, FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
//Opens an source document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(sourceStreamPath,
FormatTypeAutomatic))
{
//Sets the background type as picture
document.Background.Type = BackgroundType.Picture;
//Opens the existing image
FileStream imageStream = new FileStream(@"Data/Picture.png", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
MemoryStream memoryStream = new MemoryStream();
imageStream.CopyTo(memoryStream);
document.Background.Picture = memoryStream.ToArray();
MemoryStream stream = new MemoryStream();
//Saves and closes the destination document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}
```

XAMARIN

```
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
```

```

using (WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Hello World.docx"), FormatType.Docx))
{
document.Background.Type = BackgroundType.Picture;
//Opens the existing image
Stream imageStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Picture.png");
MemoryStream memoryStream = new MemoryStream();
imageStream.CopyTo(memoryStream);
document.Background.Picture = memoryStream.ToArray();
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWordDoc.docx", "application/msword", stream);
//Closes the document
document.Close();
//Please download the helper files from the below link to save the stream as file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
}

```

Working with Sections

A section contains the contents present in Headers, Footers and main document through the instances of **WTextBody**. A section also has a specific set of properties used to define the page settings, number of columns, headers and footers and so on that decide how the text appears. **WTextBody** represents group of paragraphs and tables etc.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company.");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to created section
Dim paragraph As IWParagraph = section.AddParagraph()
'Appends the text to the created paragraph

```

```
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new instance of WordDocument (Empty Word Document)
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new instance of WordDocument (Empty Word Document)
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
```

```
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

You can add the multiple sections into the document. When you add more than one section into the word document, the section starts from the next page by default.

You can also add a new section that starts on a same page by specifying the **BreakCode** as shown in following code example.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Adds the new section to the document
section = document.AddSection();
//Sets a section break
section.BreakCode = SectionBreakCode.NoBreak;
//Adds a paragraph to created section
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds the section into Word document
Dim section As IWSection = document.AddSection()
'Adds a paragraph to created section
```

```

Dim paragraph As IParagraph = section.AddParagraph()
Dim paraText As String = "AdventureWorks Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company."
'Appends the text to the created paragraph
paragraph.AppendText(paraText)
'Adds the new section to the document
section = document.AddSection()
'Sets a section break
section.BreakCode = SectionBreakCode.NoBreak
'Adds a paragraph to created section
paragraph = section.AddParagraph()
'Appends the text to the created paragraph
paragraph.AppendText(paraText)
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds a paragraph to created section
IParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Adds the new section to the document
section = document.AddSection();
//Sets a section break
section.BreakCode = SectionBreakCode.NoBreak;
//Adds a paragraph to created section
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds a paragraph to created section

```



```

IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Adds the new section to the document
section = document.AddSection();
//Sets a section break
section.BreakCode = SectionBreakCode.NoBreak;
//Adds a paragraph to created section
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Adds the new section to the document
section = document.AddSection();
//Sets a section break
section.BreakCode = SectionBreakCode.NoBreak;
//Adds a paragraph to created section
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Specifying Page Properties

Each section has its own page setup properties such as page size, orientation, margins, borders, etc.

The following code example shows how to set the page setup properties

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
//Sets page setup options
section.PageSetup.Orientation = PageOrientation.Landscape;
section.PageSetup.Margins.All = 72;
section.PageSetup.Borders.LineWidth = 2;
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
Dim section As IWSection = document.AddSection()
'Sets page setup options
section.PageSetup.Orientation = PageOrientation.Landscape
section.PageSetup.Margins.All = 72
section.PageSetup.Borders.LineWidth = 2
'Adds a paragraph to created section
Dim paragraph As IWParagraph = section.AddParagraph()
'Appends the text to the created paragraph.
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
//Sets page setup options
section.PageSetup.Orientation = PageOrientation.Landscape;
section.PageSetup.Margins.All = 72;
section.PageSetup.Borders.LineWidth = 2;
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to the created paragraph
```

```

paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
//Sets page setup options
section.PageSetup.Orientation = PageOrientation.Landscape;
section.PageSetup.Margins.All = 72;
section.PageSetup.Borders.LineWidth = 2;
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
//Sets page setup options
section.PageSetup.Orientation = PageOrientation.Landscape;
section.PageSetup.Margins.All = 72;
section.PageSetup.Borders.LineWidth = 2;
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream

```

```
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Creating Multi-column document

You can split the contents into two or more columns by specifying the column width and spacing between columns.

The following code example shows how to display contents in multiple columns.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
//Adds the column into the section
section.AddColumn(150, 20);
//Adds the column into the section
section.AddColumn(150, 20);
//Adds the column into the section
section.AddColumn(150, 20);
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
//Adds a paragraph to created section
paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Adds the column break
paragraph.AppendBreak(BreakType.ColumnBreak);
//Adds a paragraph to created section
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Adds the column break
paragraph.AppendBreak(BreakType.ColumnBreak);
//Adds a paragraph to created section
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
```

```

Dim document As New WordDocument()
'Adds the section into Word document
Dim section As IWSection = document.AddSection()
'Adds the column into the section
section.AddColumn(150, 20)
'Adds the column into the section
section.AddColumn(150, 20)
'Adds the column into the section
section.AddColumn(150, 20)
'Adds a paragraph to created section
Dim paragraph As IWParagraph = section.AddParagraph()
'Adds a paragraph to created section
paragraph = section.AddParagraph()
Dim paraText As String = "AdventureWorks Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company."
'Appends the text to the created paragraph
paragraph.AppendText(paraText)
'Adds the column break
paragraph.AppendBreak(BreakType.ColumnBreak)
'Adds a paragraph to created section
paragraph = section.AddParagraph()
'Appends the text to the created paragraph
paragraph.AppendText(paraText)
'Adds the column break
paragraph.AppendBreak(BreakType.ColumnBreak)
'Adds a paragraph to created section
paragraph = section.AddParagraph()
'Appends the text to the created paragraph
paragraph.AppendText(paraText)
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
//Adds the column into the section
section.AddColumn(150, 20);
//Adds the column into the section
section.AddColumn(150, 20);
//Adds the column into the section
section.AddColumn(150, 20);
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
//Adds a paragraph to created section
paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Adds the column break

```

```

paragraph.AppendBreak(BreakType.ColumnBreak);
//Adds a paragraph to created section
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Adds the column break
paragraph.AppendBreak(BreakType.ColumnBreak);
//Adds a paragraph to created section
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
//Adds the column into the section
section.AddColumn(150, 20);
//Adds the column into the section
section.AddColumn(150, 20);
//Adds the column into the section
section.AddColumn(150, 20);
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
//Adds a paragraph to created section
paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Adds the column break
paragraph.AppendBreak(BreakType.ColumnBreak);
//Adds a paragraph to created section
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Adds the column break
paragraph.AppendBreak(BreakType.ColumnBreak);
//Adds a paragraph to created section
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Saves and closes the Word document instance

```

```

MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
//Adds the column into the section
section.AddColumn(150, 20);
//Adds the column into the section
section.AddColumn(150, 20);
//Adds the column into the section
section.AddColumn(150, 20);
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
//Adds a paragraph to created section
paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Adds the column break
paragraph.AppendBreak(BreakType.ColumnBreak);
//Adds a paragraph to created section
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Adds the column break
paragraph.AppendBreak(BreakType.ColumnBreak);
//Adds a paragraph to created section
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Creating document with different page settings

You can prefer to have more sections in a Word document when you need to have different page settings or headers and footers for a specific set of contents. The following code example illustrates how to create a Word document with multiple sections whose page orientation are portrait and landscape respectively.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Sets the page orientation as portrait
section.PageSetup.Orientation = PageOrientation.Portrait;
//Adds the new section to the document
section = document.AddSection();
//Sets the section break
section.BreakCode = SectionBreakCode.NewPage;
paragraph = section.AddParagraph();
//Sets the page orientation as land scape
section.PageSetup.Orientation = PageOrientation.Landscape;
//Appends the text to the paragraph
paragraph.AppendText(paraText);
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds the section into Word document
Dim section As IWSection = document.AddSection()
'Adds a paragraph to created section
Dim paragraph As IWParagraph = section.AddParagraph()
Dim paraText As String = "AdventureWorks Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company."
'Appends the text to the created paragraph
paragraph.AppendText(paraText)
'Sets the page orientation as portrait
section.PageSetup.Orientation = PageOrientation.Portrait
'Adds the new section to the document
section = document.AddSection()
'Sets the section break
section.BreakCode = SectionBreakCode.NewPage
paragraph = section.AddParagraph()
'Sets the page orientation as landscape
section.PageSetup.Orientation = PageOrientation.Landscape
```



```
'Appends the text to the paragraph
paragraph.AppendText(paraText)
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Sets the page orientation as portrait
section.PageSetup.Orientation = PageOrientation.Portrait;
//Adds the new section to the document
section = document.AddSection();
//Sets the section break
section.BreakCode = SectionBreakCode.NewPage;
paragraph = section.AddParagraph();
//Sets the page orientation as land scape
section.PageSetup.Orientation = PageOrientation.Landscape;
//Appends the text to the paragraph
paragraph.AppendText(paraText);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Sets the page orientation as portrait
section.PageSetup.Orientation = PageOrientation.Portrait;
```

```
//Adds the new section to the document
section = document.AddSection();
//Sets the section break
section.BreakCode = SectionBreakCode.NewPage;
paragraph = section.AddParagraph();
//Sets the page orientation as landscape
section.PageSetup.Orientation = PageOrientation.Landscape;
//Appends the text to the paragraph
paragraph.AppendText(paraText);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
//Adds a paragraph to created section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends the text to the created paragraph
paragraph.AppendText(paraText);
//Sets the page orientation as portrait
section.PageSetup.Orientation = PageOrientation.Portrait;
//Adds the new section to the document
section = document.AddSection();
//Sets the section break
section.BreakCode = SectionBreakCode.NewPage;
paragraph = section.AddParagraph();
//Sets the page orientation as landscape
section.PageSetup.Orientation = PageOrientation.Landscape;
//Appends the text to the paragraph
paragraph.AppendText(paraText);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Working with Headers and Footers

Header and footer also represent the group of paragraphs and tables that occur at the top and bottom of the page respectively. Header and footer may vary for each section. The following are the types of Headers/Footers:

- FirstPageHeader – Represents the first page header of the document.
- FirstPageFooter – Represents the first page footer of the document.
- OddHeader – Represents the odd page header of the document and it is the default header for the section.
- OddFooter – Represents the odd page footer of the document and it is the default footer for the section.
- EvenHeader – Represents the even page header of the document.
- Even Footer - Represents the even page footer of the document.

The following code example illustrates how to add simple header and footer into a Word document.

C#

```
//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Default Page Header ]");
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Default Page Footer ]");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new document
Dim document As New WordDocument()
'Adds the first section to the document
Dim section As IWSection = document.AddSection()
'Adds a paragraph to the section
```

```

Dim paragraph As IParagraph = section.AddParagraph()
Dim paraText As String = "AdventureWorks Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company."
'Appends some text to the first page in document
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ First Page ] " & vbCr
& vbCr) & paraText)
paragraph.ParagraphFormat.PageBreakAfter = True
'Appends some text to the second page in document
paragraph = section.AddParagraph()
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ Second Page ] " &
vbCr & vbCr) & paraText)
paragraph.ParagraphFormat.PageBreakAfter = True
'Appends some text to the third page in document
paragraph = section.AddParagraph()
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ Third Page ] " & vbCr
& vbCr) & paraText)
'Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph()
paragraph.AppendText("[ Default Page Header ]")
'Inserts the default Page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph()
paragraph.AppendText("[ Default Page Footer ]")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Adds a paragraph to the section
IParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Default Page Header ]");
//Inserts the default Page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Default Page Footer ]");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();

```

```
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Default Page Header ]");
//Inserts the default Page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Default Page Footer ]");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company.";
```

```

//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Default Page Header ]");
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Default Page Footer ]");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

You can have a specific header and footer contents for the first page in a Word document. The following code illustrates the same.

C#

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Sets DifferentFirstPage as true for inserting header and footer text
section.PageSetup.DifferentFirstPage = true;
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the first page header

```

```

paragraph = section.HeadersFooters.FirstPageHeader.AddParagraph();
paragraph.AppendText("[First Page Header ]");
//Inserts the first page footer
paragraph = section.HeadersFooters.FirstPageFooter.AddParagraph();
paragraph.AppendText("[ First Page Footer ]");
//Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Default Page Header ]");
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Default Page Footer ]");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new document
Dim document As New WordDocument()
'Adds the first section to the document
Dim section As IWSection = document.AddSection()
'Sets DifferentFirstPage as true for inserting header and footer text
section.PageSetup.DifferentFirstPage = True
'Adds a paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
Dim paraText As String = "AdventureWorks Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company."
'Appends some text to the first page in document
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ First Page ] " & vbCr
& vbCr) & paraText)
paragraph.ParagraphFormat.PageBreakAfter = True
'Appends some text to the second page in document
paragraph = section.AddParagraph()
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ Second Page ] " &
vbCr & vbCr) & paraText)
paragraph.ParagraphFormat.PageBreakAfter = True
'Appends some text to the third page in document
paragraph = section.AddParagraph()
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ Third Page ] " & vbCr
& vbCr) & paraText)
'Inserts the first page header
paragraph = section.HeadersFooters.FirstPageHeader.AddParagraph()
paragraph.AppendText("[First Page Header ]")
'Inserts the first page footer
paragraph = section.HeadersFooters.FirstPageFooter.AddParagraph()
paragraph.AppendText("[ First Page Footer ]")
'Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph()
paragraph.AppendText("[ Default Page Header ]")
'Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph()
paragraph.AppendText("[ Default Page Footer ]")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```
//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Sets DifferentFirstPage as true for inserting header and footer text
section.PageSetup.DifferentFirstPage = true;
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the first page header
paragraph = section.HeadersFooters.FirstPageHeader.AddParagraph();
paragraph.AppendText("[First Page Header ]");
//Inserts the first page footer
paragraph = section.HeadersFooters.FirstPageFooter.AddParagraph();
paragraph.AppendText("[ First Page Footer ]");
//Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Default Page Header ]");
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Default Page Footer ]");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Sets DifferentFirstPage as true for inserting header and footer text
section.PageSetup.DifferentFirstPage = true;
//Adds a paragraph to the section
```



```

IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the first page header
paragraph = section.HeadersFooters.FirstPageHeader.AddParagraph();
paragraph.AppendText("[First Page Header ]");
//Inserts the first page footer
paragraph = section.HeadersFooters.FirstPageFooter.AddParagraph();
paragraph.AppendText("[ First Page Footer ]");
//Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Default Page Header ]");
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Default Page Footer ]");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Sets DifferentFirstPage as true for inserting header and footer text
section.PageSetup.DifferentFirstPage = true;
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document

```

```

paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the first page header
paragraph = section.HeadersFooters.FirstPageHeader.AddParagraph();
paragraph.AppendText("[First Page Header ]");
//Inserts the first page footer
paragraph = section.HeadersFooters.FirstPageFooter.AddParagraph();
paragraph.AppendText("[ First Page Footer ]");
//Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Default Page Header ]");
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Default Page Footer ]");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

A Word document can have different header and footer for odd and even pages.

The following code example shows how to set different header and footer for the odd and even pages of the document.

C#

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Sets DifferentOddAndEvenPages as true for inserting header and footer text
section.PageSetup.DifferentOddAndEvenPages = true;
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the odd page header

```

```

paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Odd Page Header ]");
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Odd Page Footer ]");
//Inserts the even page header
paragraph = section.HeadersFooters.EvenHeader.AddParagraph();
paragraph.AppendText("[Even Page Header ]");
//Inserts the even page footer
paragraph = section.HeadersFooters.EvenFooter.AddParagraph();
paragraph.AppendText("[ Even Page Footer ]");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new document
Dim document As New WordDocument()
'Adds the first section to the document
Dim section As IWSection = document.AddSection()
'Sets DifferentOddAndEvenPages as true for inserting header and footer text
section.PageSetup.DifferentOddAndEvenPages = True
'Adds a paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
Dim paraText As String = "AdventureWorks Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company."
'Appends some text to the first page in document
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ First Page ] " & vbCr
& vbCr) & paraText)
paragraph.ParagraphFormat.PageBreakAfter = True
'Appends some text to the second page in document
paragraph = section.AddParagraph()
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ Second Page ] " &
vbCr & vbCr) & paraText)
paragraph.ParagraphFormat.PageBreakAfter = True
'Appends some text to the third page in document
paragraph = section.AddParagraph()
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ Third Page ] " & vbCr
& vbCr) & paraText)
'Inserts the odd page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph()
paragraph.AppendText("[ Odd Page Header ]")
'Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph()
paragraph.AppendText("[ Odd Page Footer ]")
'Inserts the even page header
paragraph = section.HeadersFooters.EvenHeader.AddParagraph()
paragraph.AppendText("[Even Page Header ]")
'Inserts the even page footer
paragraph = section.HeadersFooters.EvenFooter.AddParagraph()
paragraph.AppendText("[ Even Page Footer ]")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Sets DifferentOddAndEvenPages as true for inserting header and footer text
section.PageSetup.DifferentOddAndEvenPages = true;
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the odd page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Odd Page Header ]");
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Odd Page Footer ]");
//Inserts the even page header
paragraph = section.HeadersFooters.EvenHeader.AddParagraph();
paragraph.AppendText("[Even Page Header ]");
//Inserts the even page footer
paragraph = section.HeadersFooters.EvenFooter.AddParagraph();
paragraph.AppendText("[ Even Page Footer ]");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Sets DifferentOddAndEvenPages as true for inserting header and footer text
section.PageSetup.DifferentOddAndEvenPages = true;
//Adds a paragraph to the section

```

```

IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the odd page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Odd Page Header ]");
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Odd Page Footer ]");
//Inserts the even page header
paragraph = section.HeadersFooters.EvenHeader.AddParagraph();
paragraph.AppendText("[Even Page Header ]");
//Inserts the even page footer
paragraph = section.HeadersFooters.EvenFooter.AddParagraph();
paragraph.AppendText("[ Even Page Footer ]");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Sets DifferentOddAndEvenPages as true for inserting header and footer text
section.PageSetup.DifferentOddAndEvenPages = true;
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document

```

```

paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the odd page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Odd Page Header ]");
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Odd Page Footer ]");
//Inserts the even page header
paragraph = section.HeadersFooters.EvenHeader.AddParagraph();
paragraph.AppendText("[Even Page Header ]");
//Inserts the even page footer
paragraph = section.HeadersFooters.EvenFooter.AddParagraph();
paragraph.AppendText("[ Even Page Footer ]");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

You can use the previous section header and footer for the current section by using `LinkToPrevious` property.

The following code example shows how to link the previous section header and footer for the current section.

C#

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Inserts the first section header
section.HeadersFooters.Header.AddParagraph().AppendText("[ First Section
Header ]");
//Inserts the first section footer
section.HeadersFooters.Footer.AddParagraph().AppendText("[ First Section
Footer ]");
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
//Adds the second section to the document
section = document.AddSection();
//Inserts the second section header

```

```

section.HeadersFooters.Header.AddParagraph().AppendText("[ Second Section
Header ]");
//Inserts the second section footer.
section.HeadersFooters.Footer.AddParagraph().AppendText("[ Second Section
Footer ]");
//Sets LinkToPrevious as true for retrieve the header and footer from
previous section
section.HeadersFooters.LinkToPrevious = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
//Adds the third section to the document
section = document.AddSection();
//Inserts the third section header
section.HeadersFooters.Header.AddParagraph().AppendText("[ Third Section
Header ]");
//Inserts the third section footer
section.HeadersFooters.Footer.AddParagraph().AppendText("[ Third Section
Footer ]");
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new document
Dim document As New WordDocument()
'Adds the first section to the document
Dim section As IWSection = document.AddSection()
'Inserts the first section header
section.HeadersFooters.Header.AddParagraph().AppendText("[ First Section
Header ]")
'Inserts the first section footer
section.HeadersFooters.Footer.AddParagraph().AppendText("[ First Section
Footer ]")
'Adds a paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
Dim paraText As String = "AdventureWorks Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company."
'Appends some text to the first page in document
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ First Page ] " & vbCr
& vbCr) & paraText)
'Adds the second section to the document
section = document.AddSection()
'Inserts the second section header
section.HeadersFooters.Header.AddParagraph().AppendText("[ Second Section
Header ]")
'Inserts the second section footer
section.HeadersFooters.Footer.AddParagraph().AppendText("[ Second Section
Footer ]")
'Sets LinkToPrevious as true for retrieve the header and footer from
previous section

```

```

section.HeadersFooters.LinkToPrevious = True
'Appends some text to the second page in document
paragraph = section.AddParagraph()
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ Second Page ] " &
vbCr & vbCr) & paraText)
'Adds the third section to the document
section = document.AddSection()
'Inserts the third section header
section.HeadersFooters.Header.AddParagraph().AppendText("[ Third Section
Header ]")
'Inserts the third section footer
section.HeadersFooters.Footer.AddParagraph().AppendText("[ Third Section
Footer ]")
'Appends some text to the third page in document
paragraph = section.AddParagraph()
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ Third Page ] " & vbCr
& vbCr) & paraText)
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Inserts the first section header
section.HeadersFooters.Header.AddParagraph().AppendText("[ First Section
Header ]");
//Inserts the first section footer
section.HeadersFooters.Footer.AddParagraph().AppendText("[ First Section
Footer ]");
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
//Adds the second section to the document
section = document.AddSection();
//Inserts the second section header
section.HeadersFooters.Header.AddParagraph().AppendText("[ Second Section
Header ]");
//Inserts the second section footer.
section.HeadersFooters.Footer.AddParagraph().AppendText("[ Second Section
Footer ]");
//Sets LinkToPrevious as true for retrieve the header and footer from
previous section
section.HeadersFooters.LinkToPrevious = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
//Adds the third section to the document
section = document.AddSection();

```



```
//Inserts the third section header
section.HeadersFooters.Header.AddParagraph().AppendText("[ Third Section
Header ]");
//Inserts the third section footer
section.HeadersFooters.Footer.AddParagraph().AppendText("[ Third Section
Footer ]");
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Inserts the first section header
section.HeadersFooters.Header.AddParagraph().AppendText("[ First Section
Header ]");
//Inserts the first section footer
section.HeadersFooters.Footer.AddParagraph().AppendText("[ First Section
Footer ]");
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
//Adds the second section to the document
section = document.AddSection();
//Inserts the second section header
section.HeadersFooters.Header.AddParagraph().AppendText("[ Second Section
Header ]");
//Inserts the second section footer.
section.HeadersFooters.Footer.AddParagraph().AppendText("[ Second Section
Footer ]");
//Sets LinkToPrevious as true for retrieve the header and footer from
previous section
section.HeadersFooters.LinkToPrevious = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
//Adds the third section to the document
section = document.AddSection();
//Inserts the third section header
```

```

section.HeadersFooters.Header.AddParagraph().AppendText("[ Third Section
Header ]");
//Inserts the third section footer
section.HeadersFooters.Footer.AddParagraph().AppendText("[ Third Section
Footer ]");
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Inserts the first section header
section.HeadersFooters.Header.AddParagraph().AppendText("[ First Section
Header ]");
//Inserts the first section footer
section.HeadersFooters.Footer.AddParagraph().AppendText("[ First Section
Footer ]");
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
//Adds the second section to the document
section = document.AddSection();
//Inserts the second section header
section.HeadersFooters.Header.AddParagraph().AppendText("[ Second Section
Header ]");
//Inserts the second section footer.
section.HeadersFooters.Footer.AddParagraph().AppendText("[ Second Section
Footer ]");
//Sets LinkToPrevious as true for retrieve the header and footer from
previous section
section.HeadersFooters.LinkToPrevious = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
//Adds the third section to the document
section = document.AddSection();
//Inserts the third section header
section.HeadersFooters.Header.AddParagraph().AppendText("[ Third Section
Header ]");
//Inserts the third section footer

```

```

section.HeadersFooters.Footer.AddParagraph().AppendText("[ Third Section
Footer ]");
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Adding Page Numbers

You can insert the current page number within the document contents. The following code example illustrates how to insert current page number within footer.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
section.PageSetup.PageStartingNumber = 1;
section.PageSetup.RestartPageNumbering = true;
section.PageSetup.PageNumberStyle = PageNumberStyle.Arabic;
//Adds a footer paragraph text to the document
IWParagraph paragraph = section.HeadersFooters.Footer.AddParagraph();
paragraph.ParagraphFormat.Tabs.AddTab(523f, TabJustification.Right,
TabLeader.NoLeader);
//Adds text for the footer paragraph
paragraph.AppendText("Copyright Northwind Inc. 2001 - 2015");
//Adds page number field to the document
paragraph.AppendText("\tPage ");
paragraph.AppendField("Page", FieldType.FieldPage);
//Adds the paragraph
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds the section into Word document

```

```

Dim section As IWSection = document.AddSection()
section.PageSetup.PageStartingNumber = 1
section.PageSetup.RestartPageNumbering = True
section.PageSetup.PageNumberStyle = PageNumberStyle.Arabic
'Adds a footer paragraph text to the document
Dim paragraph As IWParagraph = section.HeadersFooters.Footer.AddParagraph()
paragraph.ParagraphFormat.Tabs.AddTab(523.0F, TabJustification.Right,
TabLeader.NoLeader)
'Adds text for the footer paragraph
paragraph.AppendText("Copyright Northwind Inc. 2001 - 2015")
'Adds page number field to the document
paragraph.AppendText(vbTab & "Page ")
paragraph.AppendField("Page", FieldType.FieldPage)
'Adds the paragraph
paragraph = section.AddParagraph()
'Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
section.PageSetup.PageStartingNumber = 1;
section.PageSetup.RestartPageNumbering = true;
section.PageSetup.PageNumberStyle = PageNumberStyle.Arabic;
//Adds a footer paragraph text to the document
IWParagraph paragraph = section.HeadersFooters.Footer.AddParagraph();
paragraph.ParagraphFormat.Tabs.AddTab(523f, TabJustification.Right,
TabLeader.NoLeader);
//Adds text for the footer paragraph
paragraph.AppendText("Copyright Northwind Inc. 2001 - 2015");
//Adds page number field to the document
paragraph.AppendText("\tPage ");
paragraph.AppendField("Page", FieldType.FieldPage);
//Adds the paragraph
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform

```

[//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp](https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp)

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
section.PageSetup.PageStartingNumber = 1;
section.PageSetup.RestartPageNumbering = true;
section.PageSetup.PageNumberStyle = PageNumberStyle.Arabic;
//Adds a footer paragraph text to the document
IWParagraph paragraph = section.HeadersFooters.Footer.AddParagraph();
paragraph.ParagraphFormat.Tabs.AddTab(523f, TabJustification.Right,
TabLeader.NoLeader);
//Adds text for the footer paragraph
paragraph.AppendText("Copyright Northwind Inc. 2001 - 2015");
//Adds page number field to the document
paragraph.AppendText("\tPage ");
paragraph.AppendField("Page", FieldType.FieldPage);
//Adds the paragraph
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
section.PageSetup.PageStartingNumber = 1;
section.PageSetup.RestartPageNumbering = true;
section.PageSetup.PageNumberStyle = PageNumberStyle.Arabic;
//Adds a footer paragraph text to the document
IWParagraph paragraph = section.HeadersFooters.Footer.AddParagraph();
paragraph.ParagraphFormat.Tabs.AddTab(523f, TabJustification.Right,
TabLeader.NoLeader);
//Adds text for the footer paragraph
paragraph.AppendText("Copyright Northwind Inc. 2001 - 2015");
//Adds page number field to the document
paragraph.AppendText("\tPage ");
paragraph.AppendField("Page", FieldType.FieldPage);
//Adds the paragraph
paragraph = section.AddParagraph();
```

```
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

The following code example illustrates how to add the current page number and total number of pages in header/footer.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
section.PageSetup.PageStartingNumber = 1;
section.PageSetup.RestartPageNumbering = true;
section.PageSetup.PageNumberStyle = PageNumberStyle.Arabic;
//Adds a footer paragraph text to the document
IWParagraph paragraph = section.HeadersFooters.Footer.AddParagraph();
paragraph.ParagraphFormat.Tabs.AddTab(523f, TabJustification.Right,
TabLeader.NoLeader);
// Adds text for the footer paragraph
paragraph.AppendText("Copyright Northwind Inc. 2001 - 2015\t");
//Adds the text
paragraph.AppendText(" Page ");
//Adds page number field to the document
paragraph.AppendField("CurrentPageNumber", FieldType.FieldPage);
// Adds the text
paragraph.AppendText(" of ");
//Adds number of page field to the document
paragraph.AppendField("TotalNumberOfPages", FieldType.FieldNumPages);
//Adds the paragraph
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
```

```

Dim document As New WordDocument()
'Adds the section into Word document
Dim section As IWSection = document.AddSection()
section.PageSetup.PageStartingNumber = 1
section.PageSetup.RestartPageNumbering = True
section.PageSetup.PageNumberStyle = PageNumberStyle.Arabic
'Adds a footer paragraph text to the document
Dim paragraph As IWParagraph = section.HeadersFooters.Footer.AddParagraph()
paragraph.ParagraphFormat.Tabs.AddTab(523.0F, TabJustification.Right,
TabLeader.NoLeader)
'Adds text for the footer paragraph
paragraph.AppendText("Copyright Northwind Inc. 2001 - 2015" & vbTab)
'Adds the text
paragraph.AppendText(" Page ")
'Adds page number field to the document
paragraph.AppendField("CurrentPageNumber", FieldType.FieldPage)
'Adds the text
paragraph.AppendText(" of ")
'Adds number of page field to the document
paragraph.AppendField("TotalNumberOfPages", FieldType.FieldNumPages)
'Adds the paragraph
paragraph = section.AddParagraph()
'Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
section.PageSetup.PageStartingNumber = 1;
section.PageSetup.RestartPageNumbering = true;
section.PageSetup.PageNumberStyle = PageNumberStyle.Arabic;
//Adds a footer paragraph text to the document
IWParagraph paragraph = section.HeadersFooters.Footer.AddParagraph();
paragraph.ParagraphFormat.Tabs.AddTab(523f, TabJustification.Right,
TabLeader.NoLeader);
//Adds text for the footer paragraph
paragraph.AppendText("Copyright Northwind Inc. 2001 - 2015\t");
//Adds the text
paragraph.AppendText(" Page ");
//Adds page number field to the document
paragraph.AppendField("CurrentPageNumber", FieldType.FieldPage);
// Adds the text
paragraph.AppendText(" of ");
//Adds number of page field to the document
paragraph.AppendField("TotalNumberOfPages", FieldType.FieldNumPages);
//Adds the paragraph
paragraph = section.AddParagraph();
//Appends the text to the created paragraph

```

```

paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into Word document
IWSection section = document.AddSection();
section.PageSetup.PageStartingNumber = 1;
section.PageSetup.RestartPageNumbering = true;
section.PageSetup.PageNumberStyle = PageNumberStyle.Arabic;
//Adds a footer paragraph text to the document
IWParagraph paragraph = section.HeadersFooters.Footer.AddParagraph();
paragraph.ParagraphFormat.Tabs.AddTab(523f, TabJustification.Right,
TabLeader.NoLeader);
// Adds text for the footer paragraph
paragraph.AppendText("Copyright Northwind Inc. 2001 - 2015\t");
//Adds the text
paragraph.AppendText(" Page ");
//Adds page number field to the document
paragraph.AppendField("CurrentPageNumber", FieldType.FieldPage);
//Adds the text
paragraph.AppendText(" of ");
//Adds number of page field to the document
paragraph.AppendField("TotalNumberOfPages", FieldType.FieldNumPages);
//Adds the paragraph
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();

```



```

//Adds the section into Word document
IWSection section = document.AddSection();
section.PageSetup.PageStartingNumber = 1;
section.PageSetup.RestartPageNumbering = true;
section.PageSetup.PageNumberStyle = PageNumberStyle.Arabic;
//Adds a footer paragraph text to the document
IWParagraph paragraph = section.HeadersFooters.Footer.AddParagraph();
paragraph.ParagraphFormat.Tabs.AddTab(523f, TabJustification.Right,
TabLeader.NoLeader);
//Adds text for the footer paragraph
paragraph.AppendText("Copyright Northwind Inc. 2001 - 2015\t");
//Adds the text
paragraph.AppendText(" Page ");
//Adds page number field to the document
paragraph.AppendField("CurrentPageNumber", FieldType.FieldPage);
//Adds the text
paragraph.AppendText(" of ");
//Adds number of page field to the document
paragraph.AppendField("TotalNumberOfPages", FieldType.FieldNumPages);
//Adds the paragraph
paragraph = section.AddParagraph();
//Appends the text to the created paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example shows how to adjust the height of header and footer.

C#

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Specifies the value to header distance
section.PageSetup.HeaderDistance = 100;
//Specifies the value to footer distance
section.PageSetup.FooterDistance = 100;
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document

```

```

paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Default Page Header ]");
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Default Page Footer ]");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new document
Dim document As New WordDocument()
'Adds the first section to the document
Dim section As IWSection = document.AddSection()
'Specifies the value to header distance
section.PageSetup.HeaderDistance = 100
'Specifies the value to footer distance
section.PageSetup.FooterDistance = 100
'Adds a paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
Dim paraText As String = "AdventureWorks Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company."
'Appends some text to the first page in document
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ First Page ] " & vbCr
& vbCr) & paraText)
paragraph.ParagraphFormat.PageBreakAfter = True
'Appends some text to the second page in document
paragraph = section.AddParagraph()
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ Second Page ] " &
vbCr & vbCr) & paraText)
paragraph.ParagraphFormat.PageBreakAfter = True
'Appends some text to the third page in document
paragraph = section.AddParagraph()
paragraph.AppendText(Convert.ToString(vbCr & vbCr & "[ Third Page ] " & vbCr
& vbCr) & paraText)
'Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph()
paragraph.AppendText("[ Default Page Header ]")
'Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph()
paragraph.AppendText("[ Default Page Footer ]")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Specifies the value to header distance
section.PageSetup.HeaderDistance = 100;
//Specifies the value to footer distance
section.PageSetup.FooterDistance = 100;
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Default Page Header ]");
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Default Page Footer ]");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Specifies the value to header distance
section.PageSetup.HeaderDistance = 100;
//Specifies the value to footer distance
section.PageSetup.FooterDistance = 100;
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();

```

```

string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Default Page Header ]");
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Default Page Footer ]");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new document
WordDocument document = new WordDocument();
//Adds the first section to the document
IWSection section = document.AddSection();
//Specifies the value to header distance
section.PageSetup.HeaderDistance = 100;
//Specifies the value to footer distance
section.PageSetup.FooterDistance = 100;
//Adds a paragraph to the section
IWParagraph paragraph = section.AddParagraph();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Appends some text to the first page in document
paragraph.AppendText("\r\r[ First Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the second page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Second Page ] \r\r" + paraText);
paragraph.ParagraphFormat.PageBreakAfter = true;
//Appends some text to the third page in document
paragraph = section.AddParagraph();
paragraph.AppendText("\r\r[ Third Page ] \r\r" + paraText);
//Inserts the default page header
paragraph = section.HeadersFooters.OddHeader.AddParagraph();
paragraph.AppendText("[ Default Page Header ]");

```

```
//Inserts the default page footer
paragraph = section.HeadersFooters.OddFooter.AddParagraph();
paragraph.AppendText("[ Default Page Footer ]");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Removing a Section

The following code example illustrates how to remove a particular section from the Word document

C#

```
//Opens an input Word template
WordDocument document = new WordDocument(inputFileName);
//Removes the second section from the collection
document.Sections.RemoveAt(1);
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Opens an input Word template
Dim document As New WordDocument(inputFileName)
'Removes the second section from the collection
document.Sections.RemoveAt(1)
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
/"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputFileStream = assembly.GetManifestResourceStream(inputFileName);
//Opens an input Word template
WordDocument document = new WordDocument(inputFileStream);
//Removes the second section from the collection
document.Sections.RemoveAt(1);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
```

```
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
FileStream inputFileStream = new FileStream(inputFileName, FileMode.Open,
    FileAccess.ReadWrite);
//Opens an input Word template
WordDocument document = new WordDocument(inputFileStream,
    FormatType.Automatic);
//Removes the second section from the collection
document.Sections.RemoveAt(1);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputFileStream = assembly.GetManifestResourceStream(inputFileName);
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(inputFileStream,
    FormatType.Automatic);
//Removes the second section from the collection
document.Sections.RemoveAt(1);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
    "application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
```

Working with Paragraph

Paragraph is the basic element in a Word document that contains a textual and graphical contents. Each paragraph has its own formatting such as line spacing, alignment, indentation, and more. Within a paragraph, the contents are represented by one or more child elements such as **WTextRange**, **WPicture**, and **Hyperlink** and more. The **ParagraphItem** is the base class for the child elements of paragraph. The following elements can be the child elements of a paragraph:

- Text: Represented by an instance of **WTextRange**.

- Image: Represented by an instance of `WPicture`.
- Comments: Represented by an instance of `WComment`.
- Hyperlink: Represented by an instance of `Hyperlink`.
- Symbols: Represented by an instance of `WSymbol`.
- Breaks: Represented by an instance of `Break`.
- OLE Object: Represented by an instance of `WOleObject`.
- Shapes: Represented by an instance of `Shape`.
- TextBox: Represented by an instance of `WTextBox`.
- Chart: Represented by an instance of `WChart`.
- Fields: Represented by an instance of `WField`.
- Form Fields: Represented by an instance of `WFormField`.
- Bookmarks: Represented by instances of `BookmarkStart` and `BookmarkEnd`.
- Absolute Tab: Represented by an instance of `WAbsoluteTab`.
- Footnotes and Endnotes: Represented by an instance of `WFootnote`.

The following code example explains how to add a new paragraph.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new text to the paragraph
paragraph.AppendText("Adding new paragraph to the document");
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Adds new text to the paragraph
paragraph.AppendText("Adding new paragraph to the document")
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
```

```

IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new text to the paragraph
paragraph.AppendText("Adding new paragraph to the document");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new text to the paragraph
paragraph.AppendText("Adding new paragraph to the document");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new text to the paragraph
paragraph.AppendText("Adding new paragraph to the document");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform

```


[//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin](https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin)

The following code example illustrates how to modify an existing paragraph.

C#

```
//Loads the template document
WordDocument document = new WordDocument("Template.docx");
//Gets the text body of first section
WTextBody textBody = document.Sections[0].Body;
//Gets the paragraph at index 1
WParagraph paragraph = textBody.Paragraphs[1];
//Iterates through the child elements of paragraph
foreach (ParagraphItem item in paragraph.ChildEntities)
{
    if (item is WTextRange)
    {
        WTextRange text = item as WTextRange;
        //Modifies the character format of the text
        text.CharacterFormat.Bold = true;
        break;
    }
}
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Loads the template document
Dim document As New WordDocument("Template.docx")
'Gets the text body of first section
Dim textBody As WTextBody = document.Sections(0).Body
'Gets the paragraph at index 1
Dim paragraph As WParagraph = textBody.Paragraphs(1)
'Iterates through the child elements of paragraph
For Each item As ParagraphItem In paragraph.ChildEntities
    If TypeOf item Is WTextRange Then
        Dim text As WTextRange = TryCast(item, WTextRange)
        'Modifies the character format of the text
        text.CharacterFormat.Bold = True
    Exit For
    End If
Next
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
```

```

Stream fileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Template.docx");
//Loads the template document
WordDocument document = new WordDocument(fileStream);
//Gets the text body of first section
WTextBody textBody = document.Sections[0].Body;
//Gets the paragraph at index 1
WParagraph paragraph = textBody.Paragraphs[1];
//Iterates through the child elements of paragraph
foreach (ParagraphItem item in paragraph.ChildEntities)
{
    if (item is WTextRange)
    {
        WTextRange text = item as WTextRange;
        //Modifies the character format of the text
        text.CharacterFormat.Bold = true;
        break;
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

FileStream fileStream = new FileStream(@"Test.docx", FileMode.Open,
FileAccess.ReadWrite);
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatTypeAutomatic);
//Gets the text body of first section
WTextBody textBody = document.Sections[0].Body;
//Gets the paragraph at index 1
WParagraph paragraph = textBody.Paragraphs[1];
//Iterates through the child elements of paragraph
foreach (ParagraphItem item in paragraph.ChildEntities)
{
    if (item is WTextRange)
    {
        WTextRange text = item as WTextRange;
        //Modifies the character format of the text
        text.CharacterFormat.Bold = true;
        break;
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();

```

```
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Template.docx");
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatType.Automatic);
//Gets the text body of first section
WTextBody textBody = document.Sections[0].Body;
//Gets the paragraph at index 1
WParagraph paragraph = textBody.Paragraphs[1];
//Iterates through the child elements of paragraph
foreach (ParagraphItem item in paragraph.ChildEntities)
{
    if (item is WTextRange)
    {
        WTextRange text = item as WTextRange;
        //Modifies the character format of the text
        text.CharacterFormat.Bold = true;
        break;
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Applying paragraph formatting

As in the Microsoft Word, DocIO provides support for all the paragraph formatting options such as line spacing, indentation, spacing before and after, keep follow, and more. The following code example explains how to apply formatting to a paragraph.

Note: The `FirstLineIndent` can be used to update or retrieve both hanging and first line indents. Negative value for this property denotes the hanging indent and positive value denotes the first line indent of the paragraph.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
```

```

IWParagraph paragraph = section.AddParagraph();
//Adds new text to the paragraph
IWTextRange firstText = paragraph.AppendText("Paragraphs are the basic
elements of the Word document. It can contain text and images.");
//Applies paragraph formatting
paragraph.ParagraphFormat.AfterSpacing = 18f;
paragraph.ParagraphFormat.BeforeSpacing = 18f;
paragraph.ParagraphFormat.BackColor = Color.LightGray;
paragraph.ParagraphFormat.FirstLineIndent = 10f;
paragraph.ParagraphFormat.LineSpacing = 10f;
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Adds new text to the paragraph
Dim firstText As IWTextRange = paragraph.AppendText("Paragraphs are the
basic elements of the Word document. It can contain text and images.")
'Applies paragraph formatting
paragraph.ParagraphFormat.AfterSpacing = 18.0F
paragraph.ParagraphFormat.BeforeSpacing = 18.0F
paragraph.ParagraphFormat.BackColor = Color.LightGray
paragraph.ParagraphFormat.FirstLineIndent = 10.0F
paragraph.ParagraphFormat.LineSpacing = 10.0F
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new text to the paragraph
IWTextRange firstText = paragraph.AppendText("Paragraphs are the basic
elements of the Word document. It can contain text and images.");
//Applies paragraph formatting
paragraph.ParagraphFormat.AfterSpacing = 18f;
paragraph.ParagraphFormat.BeforeSpacing = 18f;
paragraph.ParagraphFormat.BackColor = Color.LightGray;
paragraph.ParagraphFormat.FirstLineIndent = 10f;
paragraph.ParagraphFormat.LineSpacing = 10f;

```

```

paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Right;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new text to the paragraph
IWTextRange firstText = paragraph.AppendText("Paragraphs are the basic
elements of the Word document. It can contain text and images.");
//Applies paragraph formatting
paragraph.ParagraphFormat.AfterSpacing = 18f;
paragraph.ParagraphFormat.BeforeSpacing = 18f;
paragraph.ParagraphFormat.BackColor = Color.LightGray;
paragraph.ParagraphFormat.FirstLineIndent = 10f;
paragraph.ParagraphFormat.LineSpacing = 10f;
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Right;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new text to the paragraph
IWTextRange firstText = paragraph.AppendText("Paragraphs are the basic
elements of the Word document. It can contain text and images.");
//Applies paragraph formatting
paragraph.ParagraphFormat.AfterSpacing = 18f;
paragraph.ParagraphFormat.BeforeSpacing = 18f;
paragraph.ParagraphFormat.BackColor = Syncfusion.Drawing.Color.LightGray;

```

```

paragraph.ParagraphFormat.FirstLineIndent = 10f;
paragraph.ParagraphFormat.LineSpacing = 10f;
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Right;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Paragraph style

Paragraph style contains definition for both font (text) and paragraph formatting that can be applied to the contents of an entire paragraph. DocIO supports various pre-defined styles and also provides ability to create custom paragraph styles.

Tips: You can define a custom style or modify any built-in style to the required formatting, and apply this style to the part of Word document to be formatted. You can reduce the file size and code length by using styles instead of formatting each element explicitly.

The following code example explains how to use the predefined styles.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds new text to the paragraph
IWTextRange firstText = firstParagraph.AppendText("Built-in styles can be
applied to the paragraph. Heading1 style is applied to this paragraph.");
//Applies built-in style for the paragraph
firstParagraph.ApplyStyle(BuiltinStyle.Heading1);
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim firstParagraph As IWParagraph = section.AddParagraph()
'Adds new text to the paragraph

```

```

Dim firstText As ITextRange = firstParagraph.AppendText("Built-in styles
can be applied to the paragraph. Heading1 style is applied to this
paragraph.")
'Applies built-in style for the paragraph
firstParagraph.ApplyStyle(BuiltinStyle.Heading1)
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds new text to the paragraph
ITextRange firstText = firstParagraph.AppendText("Built-in styles can be
applied to the paragraph. Heading1 style is applied to this paragraph.");
//Applies built-in style for the paragraph
firstParagraph.ApplyStyle(BuiltinStyle.Heading1);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds new text to the paragraph
ITextRange firstText = firstParagraph.AppendText("Built-in styles can be
applied to the paragraph. Heading1 style is applied to this paragraph.");
//Applies built-in style for the paragraph
firstParagraph.ApplyStyle(BuiltinStyle.Heading1);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds new text to the paragraph
IWTextRange firstText = firstParagraph.AppendText("Built-in styles can be
applied to the paragraph. Heading1 style is applied to this paragraph.");
//Applies built-in style for the paragraph
firstParagraph.ApplyStyle(BuiltinStyle.Heading1);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Custom paragraph style

The following code example explains how to create a custom paragraph style and apply it to a paragraph.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Creates user defined style
IWParagraphStyle style = document.AddParagraphStyle("User_defined_style");
style.ParagraphFormat.BackColor = Color.LightGray;
style.ParagraphFormat.AfterSpacing = 18f;
style.ParagraphFormat.BeforeSpacing = 18f;
style.ParagraphFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.DotDash;
style.ParagraphFormat.Borders.LineWidth = 0.5f;
style.ParagraphFormat.LineSpacing = 15f;
style.CharacterFormat.FontName = "Calibri";
style.CharacterFormat.Italic = true;
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
IWTextRange text = paragraph.AppendText("A new paragraph style is created
and is applied to this paragraph.");
//Applies the new style to paragraph
paragraph.ApplyStyle("User_defined_style");
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
```



```
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Creates user defined style
Dim style As IWParagraphStyle =
document.AddParagraphStyle("User_defined_style")
style.ParagraphFormat.BackColor = Color.LightGray
style.ParagraphFormat.AfterSpacing = 18.0F
style.ParagraphFormat.BeforeSpacing = 18.0F
style.ParagraphFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.DotDash
style.ParagraphFormat.Borders.LineWidth = 0.5F
style.ParagraphFormat.LineSpacing = 15.0F
style.CharacterFormat.FontName = "Calibri"
style.CharacterFormat.Italic = True
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
Dim text As IWTextRange = paragraph.AppendText("A new paragraph style is
created and is applied to this paragraph.")
'Applies the new style to paragraph
paragraph.ApplyStyle("User_defined_style")
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Creates user defined style
IWParagraphStyle style = document.AddParagraphStyle("User_defined_style");
style.ParagraphFormat.BackColor = Color.LightGray;
style.ParagraphFormat.AfterSpacing = 18f;
style.ParagraphFormat.BeforeSpacing = 18f;
style.ParagraphFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.DotDash;
style.ParagraphFormat.Borders.LineWidth = 0.5f;
style.ParagraphFormat.LineSpacing = 15f;
style.CharacterFormat.FontName = "Calibri";
style.CharacterFormat.Italic = true;
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
IWTextRange text = paragraph.AppendText("A new paragraph style is created
and is applied to this paragraph.");
//Applies the new style to paragraph
paragraph.ApplyStyle("User_defined_style");
//Saves and closes the Word document instance
```

```

MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Creates user defined style
IWParagraphStyle style = document.AddParagraphStyle("User_defined_style");
style.ParagraphFormat.BackColor = Color.LightGray;
style.ParagraphFormat.AfterSpacing = 18f;
style.ParagraphFormat.BeforeSpacing = 18f;
style.ParagraphFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.DotDash;
style.ParagraphFormat.Borders.LineWidth = 0.5f;
style.ParagraphFormat.LineSpacing = 15f;
style.CharacterFormat.FontName = "Calibri";
style.CharacterFormat.Italic = true;
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
IWTextRange text = paragraph.AppendText("A new paragraph style is created
and is applied to this paragraph.");
//Applies the new style to paragraph
paragraph.ApplyStyle("User_defined_style");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Creates user defined style
IWParagraphStyle style = document.AddParagraphStyle("User_defined_style");
style.ParagraphFormat.BackColor = Syncfusion.Drawing.Color.LightGray;
style.ParagraphFormat.AfterSpacing = 18f;
style.ParagraphFormat.BeforeSpacing = 18f;
style.ParagraphFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.DotDash;
style.ParagraphFormat.Borders.LineWidth = 0.5f;

```

```

style.ParagraphFormat.LineSpacing = 15f;
style.CharacterFormat.FontName = "Calibri";
style.CharacterFormat.Italic = true;
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
IWTextRange text = paragraph.AppendText("A new paragraph style is created
and is applied to this paragraph.");
//Applies the new style to paragraph
paragraph.ApplyStyle("User_defined_style");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Tab stop

A tab stop is a horizontal position that is set for aligning text of the paragraph. A tab character causes the carriage to go to the next tab stop.

Each paragraph has its own tab stop collection where the new tab stop can be added and existing tab stop can be removed.

The following code example explains how to add tab stops to the paragraph.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds tab stop at position 11
Tab firstTab = paragraph.ParagraphFormat.Tabs.AddTab(11,
TabJustification.Left, TabLeader.Dotted);
//Adds tab stop at position 62
paragraph.ParagraphFormat.Tabs.AddTab(62, TabJustification.Left,
TabLeader.Single);
paragraph.AppendText("This sample\t illustrates the use of tabs in the
paragraph. Tabs\t can be inserted or removed from the paragraph.");
//Removes tab stop from the collection
paragraph.ParagraphFormat.Tabs.RemoveByTabPosition(11);
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Adds tab stop at position 11
Dim firstTab As Tab = paragraph.ParagraphFormat.Tabs.AddTab(11,
TabJustification.Left, TabLeader.Dotted)
'Adds tab stop at position 62
paragraph.ParagraphFormat.Tabs.AddTab(62, TabJustification.Left,
TabLeader.[Single])
paragraph.AppendText("This sample" & vbTab & " illustrates the use of tabs
in the paragraph. Tabs" & vbTab & " can be inserted or removed from the
paragraph.")
'Removes tab stop from the collection
paragraph.ParagraphFormat.Tabs.RemoveByTabPosition(11)
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds tab stop at position 11
Tab firstTab = paragraph.ParagraphFormat.Tabs.AddTab(11,
TabJustification.Left, TabLeader.Dotted);
//Adds tab stop at position 62
paragraph.ParagraphFormat.Tabs.AddTab(62, TabJustification.Left,
TabLeader.Single);
paragraph.AppendText("This sample\t illustrates the use of tabs in the
paragraph. Tabs\t can be inserted or removed from the paragraph.");
//Removes tab stop from the collection
paragraph.ParagraphFormat.Tabs.RemoveByTabPosition(11);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document

```

```

IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds tab stop at position 11
Tab firstTab = paragraph.ParagraphFormat.Tabs.AddTab(11,
TabJustification.Left, TabLeader.Dotted);
//Adds tab stop at position 62
paragraph.ParagraphFormat.Tabs.AddTab(62, TabJustification.Left,
TabLeader.Single);
paragraph.AppendText("This sample\t illustrates the use of tabs in the
paragraph. Tabs\t can be inserted or removed from the paragraph.");
//Removes tab stop from the collection
paragraph.ParagraphFormat.Tabs.RemoveByTabPosition(11);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds tab stop at position 11
Tab firstTab = paragraph.ParagraphFormat.Tabs.AddTab(11,
TabJustification.Left, TabLeader.Dotted);
//Adds tab stop at position 62
paragraph.ParagraphFormat.Tabs.AddTab(62, TabJustification.Left,
TabLeader.Single);
paragraph.AppendText("This sample\t illustrates the use of tabs in the
paragraph. Tabs\t can be inserted or removed from the paragraph.");
//Removes tab stop from the collection
paragraph.ParagraphFormat.Tabs.RemoveByTabPosition(11);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

RTL paragraph

You can set RTL (Right-to-left) direction to the paragraph in a Word document. The following code example shows how to set RTL (Right-to-left) for a paragraph in Word document.

C#

```
//Loads the template document
WordDocument document = new WordDocument("Template.docx");
//Gets the text body of first section
WTextBody textBody = document.Sections[0].Body;
//Gets the paragraph at index 1
WParagraph paragraph = textBody.Paragraphs[1];
//Gets a value indicating whether the paragraph is right-to-left. True
indicates the paragraph direction is RTL
bool isRTL = paragraph.ParagraphFormat.Bidi;
//Sets RTL direction for a paragraph
if(!isRTL)
paragraph.ParagraphFormat.Bidi = true;
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Loads the template document
Dim document As WordDocument = New WordDocument("Template.docx")
'Gets the text body of first section
Dim textBody As WTextBody = document.Sections(0).Body
'Gets the paragraph at index 1
Dim paragraph As WParagraph = textBody.Paragraphs(1)
'Gets a value indicating whether the paragraph is right-to-left. True
indicates the paragraph direction is RTL
Dim isRTL As Boolean = paragraph.ParagraphFormat.Bidi
'Sets RTL direction for a paragraph
If Not isRTL Then
paragraph.ParagraphFormat.Bidi = True
End If
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
///"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Loads the template document
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx
"), FormatType.Docx);
//Gets the text body of first section
WTextBody textBody = document.Sections[0].Body;
//Gets the paragraph at index 1
WParagraph paragraph = textBody.Paragraphs[1];
```

```

//Gets a value indicating whether the paragraph is right-to-left. True indicates the paragraph direction is RTL
bool isRTL = paragraph.ParagraphFormat.Bidi;
//Sets RTL direction for a paragraph
if(!isRTL)
    paragraph.ParagraphFormat.Bidi = true;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
//Loads or opens an existing Word document through Open method of
WordDocument class
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Gets the text body of first section
WTextBody textBody = document.Sections[0].Body;
//Gets the paragraph at index 1
WParagraph paragraph = textBody.Paragraphs[1];
//Gets a value indicating whether the paragraph is right-to-left. True indicates the paragraph direction is RTL
bool isRTL = paragraph.ParagraphFormat.Bidi;
//Sets RTL direction for a paragraph
if(!isRTL)
    paragraph.ParagraphFormat.Bidi = true;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Loads or opens an existing Word document through Open method of
WordDocument class
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx"), FormatType.Docx);
//Gets the text body of first section
WTextBody textBody = document.Sections[0].Body;
//Gets the paragraph at index 1
WParagraph paragraph = textBody.Paragraphs[1];

```

```
//Gets a value indicating whether the paragraph is right-to-left. True
indicates the paragraph direction is RTL
bool isRTL = paragraph.ParagraphFormat.Bidi;
//Sets RTL direction for a paragraph
if(!isRTL)
paragraph.ParagraphFormat.Bidi = true;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Working with text

Text within a paragraph is represented by one or more instances of the `WTextRange`. Each `WTextRange` instance can have its own font (text) formatting.

The following code example explains how to append text to the paragraph.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds new text to the paragraph
IWTextRange firstText = firstParagraph.AppendText("A new text is added to
the paragraph.");
firstText.CharacterFormat.FontSize = 14;
firstText.CharacterFormat.Bold = true;
firstText.CharacterFormat.TextColor = Color.Green;
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim firstParagraph As IWParagraph = section.AddParagraph()
'Adds new text to the paragraph
Dim text As IWTextRange = firstParagraph.AppendText("A new text is added to
the paragraph.")
```



```

text.CharacterFormat.FontSize = 14
text.CharacterFormat.Bold = True
text.CharacterFormat.TextColor = Color.Green
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds new text to the paragraph
IWTextRange firstText = firstParagraph.AppendText("A new text is added to
the paragraph.");
firstText.CharacterFormat.FontSize = 14;
firstText.CharacterFormat.Bold = true;
firstText.CharacterFormat.TextColor = Color.Green;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds new text to the paragraph
IWTextRange firstText = firstParagraph.AppendText("A new text is added to
the paragraph.");
firstText.CharacterFormat.FontSize = 14;
firstText.CharacterFormat.Bold = true;
firstText.CharacterFormat.TextColor = Color.Green;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds new text to the paragraph
IWTextRange firstText = firstParagraph.AppendText("A new text is added to
the paragraph.");
firstText.CharacterFormat.FontSize = 14;
firstText.CharacterFormat.Bold = true;
firstText.CharacterFormat.TextColor = Syncfusion.Drawing.Color.Green;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Text in the paragraph can be modified or replaced with a new text. This can be achieved by iterating through the paragraph items.

The following code example explains how to replace the text of a text range.

C#

```
//Loads the template document
WordDocument document = new WordDocument("Template.docx");
//Gets the last paragraph
WParagraph lastParagraph = document.LastParagraph;
//Iterates through the paragraph items to get the text range and modifies
its content.
for (int i = 0; i < lastParagraph.ChildEntities.Count; i++)
{
    if (lastParagraph.ChildEntities[i] is WTextRange)
    {
        WTextRange textRange = lastParagraph.ChildEntities[i] as WTextRange;
        textRange.Text = "First text range of the last paragraph is replaced";
        textRange.CharacterFormat.FontSize = 14;
        break;
    }
}
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx")
'Gets the last paragraph
Dim lastParagraph As WParagraph = document.LastParagraph
'Iterates through the paragraph items to get the text range and modifies its content
For i As Integer = 0 To lastParagraph.ChildEntities.Count - 1
If TypeOf lastParagraph.ChildEntities(i) Is WTextRange Then
Dim textRange As WTextRange = TryCast(lastParagraph.ChildEntities(i), WTextRange)
textRange.Text = "First text range of the last paragraph is replaced"
textRange.CharacterFormat.FontSize = 14
Exit For
End If
Next
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream FileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Template.docx");
//Loads the template document
WordDocument document = new WordDocument(FileStream);
//Gets the last paragraph
WParagraph lastParagraph = document.LastParagraph;
//Iterates through the paragraph items to get the text range and modifies its content.
for (int i = 0; i < lastParagraph.ChildEntities.Count; i++)
{
if (lastParagraph.ChildEntities[i] is WTextRange)
{
WTextRange textRange = lastParagraph.ChildEntities[i] as WTextRange;
textRange.Text = "First text range of the last paragraph is replaced";
textRange.CharacterFormat.FontSize = 14;
break;
}
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

FileStream fileStream = new FileStream(@"Template.docx", FileMode.Open,
    FileAccess.ReadWrite);
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatType.Automatic);
//Gets the last paragraph
WParagraph lastParagraph = document.LastParagraph;
//Iterates through the paragraph items to get the text range and modifies
its content.
for (int i = 0; i < lastParagraph.ChildEntities.Count; i++)
{
    if (lastParagraph.ChildEntities[i] is WTextRange)
    {
        WTextRange textRange = lastParagraph.ChildEntities[i] as WTextRange;
        textRange.Text = "First text range of the last paragraph is replaced";
        textRange.CharacterFormat.FontSize = 14;
        break;
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream FileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Template.docx");
//Loads the template document
WordDocument document = new WordDocument(FileStream, FormatType.Automatic);
//Gets the last paragraph
WParagraph lastParagraph = document.LastParagraph;
//Iterates through the paragraph items to get the text range and modifies
its content.
for (int i = 0; i < lastParagraph.ChildEntities.Count; i++)
{
    if (lastParagraph.ChildEntities[i] is WTextRange)
    {
        WTextRange textRange = lastParagraph.ChildEntities[i] as WTextRange;
        textRange.Text = "First text range of the last paragraph is replaced";
        textRange.CharacterFormat.FontSize = 14;
        break;
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
    "application/msword", stream);

```

```
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Text formatting enhances the appearance of text in the document. Text formatting includes font size, font color, font name, bold, italic, underline, etc.

The following code example explains how to apply formatting to the text.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds new text to the paragraph
IWTextRange firstText = firstParagraph.AppendText("This is the first text
range. ");
//Applies formatting for first text range
firstText.CharacterFormat.Bold = true;
firstText.CharacterFormat.FontSize = 14;
firstText.CharacterFormat.Shadow = true;
firstText.CharacterFormat.SmallCaps = true;
IWTextRange secondText = firstParagraph.AppendText("This the second text
range");
//Applies formatting for second text range
secondText.CharacterFormat.HighlightColor = Color.GreenYellow;
secondText.CharacterFormat.UnderlineStyle = UnderlineStyle.DotDash;
secondText.CharacterFormat.Italic = true;
secondText.CharacterFormat.FontName = "Times New Roman";
secondText.CharacterFormat.TextColor = Color.Green;
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim firstParagraph As IWParagraph = section.AddParagraph()
'Adds new text to the paragraph
Dim firstText As IWTextRange = firstParagraph.AppendText("This is the first
text range. ")
'Applies formatting for first text range
firstText.CharacterFormat.Bold = True
firstText.CharacterFormat.FontSize = 14
firstText.CharacterFormat.Shadow = True
firstText.CharacterFormat.SmallCaps = True
```

```

Dim secondText As ITextRange = firstParagraph.AppendText("This the second
text range")
'Applies formatting for second text range
secondText.CharacterFormat.HighlightColor = Color.GreenYellow
secondText.CharacterFormat.UnderlineStyle = UnderlineStyle.DotDash
secondText.CharacterFormat.Italic = True
secondText.CharacterFormat.FontName = "Times New Roman"
secondText.CharacterFormat.TextColor = Color.Green
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds new text to the paragraph
ITextRange firstText = firstParagraph.AppendText("This is the first text
range. ");
//Applies formatting for first text range
firstText.CharacterFormat.Bold = true;
firstText.CharacterFormat.FontSize = 14;
firstText.CharacterFormat.Shadow = true;
firstText.CharacterFormat.SmallCaps = true;
ITextRange secondText = firstParagraph.AppendText("This the second text
range");
//Applies formatting for second text range
secondText.CharacterFormat.HighlightColor = Color.GreenYellow;
secondText.CharacterFormat.UnderlineStyle = UnderlineStyle.DotDash;
secondText.CharacterFormat.Italic = true;
secondText.CharacterFormat.FontName = "Times New Roman";
secondText.CharacterFormat.TextColor = Color.Green;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();

```

```
//Adds new text to the paragraph
IWTextRange firstText = firstParagraph.AppendText("This is the first text
range. ");
//Applies formatting for first text range
firstText.CharacterFormat.Bold = true;
firstText.CharacterFormat.FontSize = 14;
firstText.CharacterFormat.Shadow = true;
firstText.CharacterFormat.SmallCaps = true;
IWTextRange secondText = firstParagraph.AppendText("This the second text
range");
//Applies formatting for second text range
secondText.CharacterFormat.HighlightColor = Color.GreenYellow;
secondText.CharacterFormat.UnderlineStyle = UnderlineStyle.DotDash;
secondText.CharacterFormat.Italic = true;
secondText.CharacterFormat.FontName = "Times New Roman";
secondText.CharacterFormat.TextColor = Color.Green;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds new text to the paragraph
IWTextRange firstText = firstParagraph.AppendText("This is the first text
range. ");
//Applies formatting for first text range
firstText.CharacterFormat.Bold = true;
firstText.CharacterFormat.FontSize = 14;
firstText.CharacterFormat.Shadow = true;
firstText.CharacterFormat.SmallCaps = true;
IWTextRange secondText = firstParagraph.AppendText("This the second text
range");
//Applies formatting for second text range
secondText.CharacterFormat.HighlightColor =
Syncfusion.Drawing.Color.GreenYellow;
secondText.CharacterFormat.UnderlineStyle =
Syncfusion.Drawing.UnderlineStyle.DotDash;
secondText.CharacterFormat.Italic = true;
secondText.CharacterFormat.FontName = "Times New Roman";
secondText.CharacterFormat.TextColor = Syncfusion.Drawing.Color.Green;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
```

```
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Working with Images

DocIO provides support for both inline and absolute positioned images.

- Inline images: The position of the image is constrained to the lines of text on the page.
- Absolute positioned images: The images can be positioned anywhere irrespective of the lines of text.

The following code example explains how to add image to the paragraph.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds image to the paragraph
IWPicture picture =
firstParagraph.AppendPicture(Image.FromFile("Image.png"));
//Sets height and width for the image
picture.Height = 100;
picture.Width = 100;
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim firstParagraph As IWParagraph = section.AddParagraph()
'Adds image to the paragraph
Dim picture As IWPicture =
firstParagraph.AppendPicture(Image.FromFile("Image.png"))
'Sets height and width for the image
picture.Height = 100
picture.Width = 100
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```


UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds image to the paragraph
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream imageStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Image.png");
IWPicture picture = firstParagraph.AppendPicture(imageStream);
//Sets height and width for the image
picture.Height = 100;
picture.Width = 100;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds image to the paragraph
FileStream imageStream = new FileStream(@"Image.png", FileMode.Open,
FileAccess.ReadWrite);
IWPicture picture = firstParagraph.AppendPicture(imageStream);
//Sets height and width for the image
picture.Height = 100;
picture.Width = 100;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
```

```

WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph firstParagraph = section.AddParagraph();
//Adds image to the paragraph
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream imageStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Image.png");
IWPicture picture = firstParagraph.AppendPicture(imageStream);
//Sets height and width for the image
picture.Height = 100;
picture.Width = 100;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Image present in the document can be replaced with a new image. This can be achieved by iterating through the paragraph items.

The following code example explains how to replace an existing image.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Template.docx");
WTextBody textbody = document.Sections[0].Body;
//Iterates through the paragraphs of the textbody
foreach (WParagraph paragraph in textbody.Paragraphs)
{
    //Iterates through the child elements of paragraph
    foreach (ParagraphItem item in paragraph.ChildEntities)
    {
        if (item is WPicture)
        {
            WPicture picture = item as WPicture;
            //Replaces the image
            if (picture.Title == "Bookmark")
                picture.LoadImage(Image.FromFile("Image.png"));
        }
    }
}
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx")
Dim textbody As WTextBody = document.Sections(0).Body
'Iterates through the paragraphs of the textbody
For Each paragraph As WParagraph In textbody.Paragraphs
'Iterates through the child elements of paragraph
For Each item As ParagraphItem In paragraph.ChildEntities
If TypeOf item Is WPicture Then
Dim picture As WPicture = TryCast(item, WPicture)
'Replaces the image
If picture.Title = "Bookmark" Then
picture.LoadImage(Image.FromFile("Image.png"))
End If
End If
Next
Next
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Template.docx");
//Loads the template document
WordDocument document = new WordDocument(fileStream);
WTextBody textbody = document.Sections[0].Body;
//Iterates through the paragraphs of the textbody
foreach (WParagraph paragraph in textbody.Paragraphs)
{
//Iterates through the child elements of paragraph
foreach (ParagraphItem item in paragraph.ChildEntities)
{
if (item is WPicture)
{
WPicture picture = item as WPicture;
//Replaces the image
if (picture.Title == "Bookmark")
{
Stream imagestream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Image.png");
picture.LoadImage(imagestream);
}
}
}
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine

```

```
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
FileStream fileStream = new FileStream(@"Template.docx", FileMode.Open,
    FileAccess.ReadWrite);
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatType.Automatic);
WTextBody textbody = document.Sections[0].Body;
//Iterates through the paragraphs of the textbody
foreach (WParagraph paragraph in textbody.Paragraphs)
{
    //Iterates through the child elements of paragraph
    foreach (ParagraphItem item in paragraph.ChildEntities)
    {
        if (item is WPicture)
        {
            WPicture picture = item as WPicture;
            //Replaces the image
            if (picture.Title == "Bookmark")
            {
                FileStream imageStream = new FileStream(@"Image.png", FileMode.Open,
                    FileAccess.ReadWrite);
                picture.LoadImage(imageStream);
            }
        }
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
    assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Template.docx");
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatType.Automatic);
WTextBody textbody = document.Sections[0].Body;
//Iterates through the paragraphs of the textbody
foreach (WParagraph paragraph in textbody.Paragraphs)
{
    //Iterates through the child elements of paragraph
    foreach (ParagraphItem item in paragraph.ChildEntities)
    {
        if (item is WPicture)
        {
```

```

WPicture picture = item as WPicture;
//Replaces the image
if (picture.Title == "Bookmark")
{
    Stream imageStream =
    assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Dummy-
    Images.jpg");
    picture.LoadImage(imageStream);
}
}
}
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Images can be removed from the document by removing it from the paragraph items.

The following code example explains how to remove the image from the paragraph items.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Template.docx");
WTextBody textbody = document.Sections[0].Body;
//Iterates through the paragraphs of the textbody
foreach (WParagraph paragraph in textbody.Paragraphs)
{
    //Iterates through the child elements of paragraph
    for (int i = 0; i < paragraph.ChildEntities.Count; i++)
    {
        //Removes images from the paragraph
        if (paragraph.ChildEntities[i] is WPicture)
        {
            paragraph.Items.RemoveAt(i);
            i--;
        }
    }
}
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```
'Loads the template document
```

```

Dim document As New WordDocument("Template.docx")
Dim textbody As WTextBody = document.Sections(0).Body
'Iterates through the paragraphs of the textbody
For Each paragraph As WParagraph In textbody.Paragraphs
'Iterates through the child elements of paragraph
For i As Integer = 0 To paragraph.ChildEntities.Count - 1
'Removes images from the paragraph
If TypeOf paragraph.ChildEntities(i) Is WPicture Then
    paragraph.Items.RemoveAt(i)
    i -= 1
End If
Next
Next
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Template.docx");
//Loads the template document
WordDocument document = new WordDocument(fileStream);
WTextBody textbody = document.Sections[0].Body;
//Iterates through the paragraphs of the textbody
foreach (WParagraph paragraph in textbody.Paragraphs)
{
//Iterates through the child elements of paragraph
for (int i = 0; i < paragraph.ChildEntities.Count; i++)
{
//Removes images from the paragraph
if (paragraph.ChildEntities[i] is WPicture)
{
    paragraph.Items.RemoveAt(i);
    i--;
}
}
}

//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

FileStream fileStream = new FileStream(@"Template.docx", FileMode.Open,
FileAccess.ReadWrite);
//Loads the template document

```

```

WordDocument document = new WordDocument(fileStream, FormatTypeAutomatic);
WTextBody textbody = document.Sections[0].Body;
//Iterates through the paragraphs of the textbody
foreach (WParagraph paragraph in textbody.Paragraphs)
{
    //Iterates through the child elements of paragraph
    for (int i = 0; i < paragraph.ChildEntities.Count; i++)
    {
        //Removes images from the paragraph
        if (paragraph.ChildEntities[i] is WPicture)
        {
            paragraph.Items.RemoveAt(i);
            i--;
        }
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Template.docx");
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatTypeAutomatic);
WTextBody textbody = document.Sections[0].Body;
//Iterates through the paragraphs of the textbody
foreach (WParagraph paragraph in textbody.Paragraphs)
{
    //Iterates through the child elements of paragraph
    for (int i = 0; i < paragraph.ChildEntities.Count; i++)
    {
        //Removes images from the paragraph
        if (paragraph.ChildEntities[i] is WPicture)
        {
            paragraph.Items.RemoveAt(i);
            i--;
        }
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);

```

```
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Format and rotate images

Absolute positioned images have properties such as position, wrap formats, and alignments. These properties are not applicable when the text wrapping style is inline. You can also rotate an image and apply flipping (horizontal and vertical) to it.

The following code example explains how various picture formats can be applied to the picture.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("This paragraph has picture. ");
//Appends new picture to the paragraph
WPicture picture = paragraph.AppendPicture(Image.FromFile("Image.png")) as
WPicture;
//Sets text wrapping style - When the wrapping style is inline, the images
are not absolutely positioned. It is added next to the text range.
picture.TextWrappingStyle = TextWrappingStyle.Square;
//Sets horizontal and vertical origin
picture.HorizontalOrigin = HorizontalOrigin.Page;
picture.VerticalOrigin = VerticalOrigin.Paragraph;
//Sets width and height for the paragraph
picture.Width = 150;
picture.Height = 100;
//Sets horizontal and vertical position for the picture
picture.HorizontalPosition = 200;
picture.VerticalPosition = 150;
picture.Name = "PictureName";
//Sets horizontal and vertical alignments
picture.HorizontalAlignment = ShapeHorizontalAlignment.Center;
picture.VerticalAlignment = ShapeVerticalAlignment.Bottom;
//Sets 90 degree rotation
picture.Rotation = 90;
//Sets horizontal flip
picture.FlipHorizontal = true;
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
```



```

Dim paragraph As IWPParagraph = section.AddParagraph()
paragraph.AppendText("This paragraph has picture. ")
'Appends new picture to the paragraph
Dim picture As WPicture =
TryCast(paragraph.AppendPicture(Image.FromFile("Image.png")), WPicture)
'Sets text wrapping style - When the wrapping style is inline, the images
are not absolutely positioned. It is added next to the text range.
picture.TextWrappingStyle = TextWrappingStyle.Square
'Sets horizontal and vertical origin
picture.HorizontalOrigin = HorizontalOrigin.Page
picture.VerticalOrigin = VerticalOrigin.Paragraph
'Sets width and height for the paragraph
picture.Width = 150
picture.Height = 100
'Sets horizontal and vertical position for the picture
picture.HorizontalPosition = 200
picture.VerticalPosition = 150
picture.Name = "PictureName"
'Sets horizontal and vertical alignments
picture.HorizontalAlignment = ShapeHorizontalAlignment.Center
picture.VerticalAlignment = ShapeVerticalAlignment.Bottom
'Sets 90 degree rotation
picture.Rotation = 90
'Sets horizontal flip
picture.FlipHorizontal = true
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWPParagraph paragraph = section.AddParagraph();
paragraph.AppendText("This paragraph has picture. ");
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream imageStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Image.png");
//Appends new picture to the paragraph
WPicture picture = paragraph.AppendPicture(imageStream) as WPicture;
//Sets text wrapping style - When the wrapping style is inline, the images
are not absolutely positioned. It is added next to the text range.
picture.TextWrappingStyle = TextWrappingStyle.Square;
//Sets horizontal and vertical origin
picture.HorizontalOrigin = HorizontalOrigin.Page;
picture.VerticalOrigin = VerticalOrigin.Paragraph;
//Sets width and height for the paragraph
picture.Width = 150;
picture.Height = 100;
//Sets horizontal and vertical position for the picture
picture.HorizontalPosition = 200;
picture.VerticalPosition = 150;

```

```

picture.Name = "PictureName";
//Sets horizontal and vertical alignments
picture.HorizontalAlignment = ShapeHorizontalAlignment.Center;
picture.VerticalAlignment = ShapeVerticalAlignment.Bottom;
//Sets 90 degree rotation
picture.Rotation = 90;
//Sets horizontal flip
picture.FlipHorizontal = true;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("This paragraph has picture. ");
FileStream imageStream = new FileStream(@"Image.png", FileMode.Open,
FileAccess.ReadWrite);
//Appends new picture to the paragraph
WPicture picture = paragraph.AppendPicture(imageStream) as WPicture;
//Sets text wrapping style - When the wrapping style is inline, the images
are not absolutely positioned. It is added next to the text range.
picture.TextWrappingStyle = TextWrappingStyle.Square;
//Sets horizontal and vertical origin
picture.HorizontalOrigin = HorizontalOrigin.Page;
picture.VerticalOrigin = VerticalOrigin.Paragraph;
//Sets width and height for the paragraph
picture.Width = 150;
picture.Height = 100;
//Sets horizontal and vertical position for the picture
picture.HorizontalPosition = 200;
picture.VerticalPosition = 150;
picture.Name = "PictureName";
//Sets horizontal and vertical alignments
picture.HorizontalAlignment = ShapeHorizontalAlignment.Center;
picture.VerticalAlignment = ShapeVerticalAlignment.Bottom;
//Sets 90 degree rotation
picture.Rotation = 90;
//Sets horizontal flip
picture.FlipHorizontal = true;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);

```

```
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("This paragraph has picture. ");
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream imageStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Image.png");
//Appends new picture to the paragraph
WPicture picture = paragraph.AppendPicture(imageStream) as WPicture;
//Sets text wrapping style - When the wrapping style is inline, the images
are not absolutely positioned. It is added next to the text range.
picture.TextWrappingStyle = TextWrappingStyle.Square;
//Sets horizontal and vertical origin
picture.HorizontalOrigin = HorizontalOrigin.Page;
picture.VerticalOrigin = VerticalOrigin.Paragraph;
//Sets width and height for the paragraph
picture.Width = 150;
picture.Height = 100;
//Sets horizontal and vertical position for the picture
picture.HorizontalPosition = 200;
picture.VerticalPosition = 150;
picture.Name = "PictureName";
//Sets horizontal and vertical alignments
picture.HorizontalAlignment = ShapeHorizontalAlignment.Center;
picture.VerticalAlignment = ShapeVerticalAlignment.Bottom;
//Sets 90 degree rotation
picture.Rotation = 90;
//Sets horizontal flip
picture.FlipHorizontal = true;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

An Image with a specific title can be retrieved by iterating the paragraph items that can be used for further manipulations.

The following code example explains how images can be iterated from the document elements.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument("Template.docx");
//Gets textbody content
WTextBody textBody = document.Sections[0].Body;
//Iterates through the textbody child entities
foreach (TextBodyItem item in textBody.ChildEntities)
{
    if (item is WParagraph)
    {
        WParagraph paragraph = item as WParagraph;
        foreach (ParagraphItem paraItem in paragraph.ChildEntities)
        {
            //Gets the image from its title and modifies its width and height
            if (paraItem is WPicture)
            {
                WPicture picture = paraItem as WPicture;
                if (picture.Title == "Bookmark")
                {
                    picture.Width = 150;
                    picture.Height = 100;
                }
            }
        }
    }
}
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument("Template.docx")
'Gets textbody content
Dim textBody As WTextBody = document.Sections(0).Body
'Iterates through the textbody child entities
For Each item As TextBodyItem In textBody.ChildEntities
    If TypeOf item Is WParagraph Then
        Dim paragraph As WParagraph = TryCast(item, WParagraph)
        For Each paraItem As ParagraphItem In paragraph.ChildEntities
            'Gets the image from its title and modifies its width and height
            If TypeOf paraItem Is WPicture Then
                Dim picture As WPicture = TryCast(paraItem, WPicture)
                If picture.Title = "Bookmark" Then
                    picture.Width = 150
                    picture.Height = 100
                End If
            End If
        Next
    End If
Next
'Saves the Word document
```

```
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Template.docx");
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(fileStream);
//Gets textbody content
WTextBody textBody = document.Sections[0].Body;
//Iterates through the textbody child entities
foreach (TextBodyItem item in textBody.ChildEntities)
{
    if (item is WParagraph)
    {
        WParagraph paragraph = item as WParagraph;
        foreach (ParagraphItem paraItem in paragraph.ChildEntities)
        {
            //Gets the image from its title and modifies its width and height
            if (paraItem is WPicture)
            {
                WPicture picture = paraItem as WPicture;
                if (picture.Title == "Bookmark")
                {
                    picture.Width = 150;
                    picture.Height = 100;
                }
            }
        }
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
FileStream fileStream = new FileStream(@"Template.docx", FileMode.Open,
FileAccess.ReadWrite);
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(fileStream, FormatTypeAutomatic);
//Gets textbody content
WTextBody textBody = document.Sections[0].Body;
//Iterates through the textbody child entities
foreach (TextBodyItem item in textBody.ChildEntities)
{
```

```

if (item is WParagraph)
{
    WParagraph paragraph = item as WParagraph;
    foreach (ParagraphItem paraItem in paragraph.ChildEntities)
    {
        //Gets the image from its title and modifies its width and height
        if (paraItem is WPicture)
        {
            WPicture picture = paraItem as WPicture;
            if (picture.Title == "Bookmark")
            {
                picture.Width = 150;
                picture.Height = 100;
            }
        }
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Template.docx");
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument(fileStream, FormatTypeAutomatic);
//Gets textbody content
WTextBody textBody = document.Sections[0].Body;
//Iterates through the textbody child entities
foreach (TextBodyItem item in textBody.ChildEntities)
{
    if (item is WParagraph)
    {
        WParagraph paragraph = item as WParagraph;
        foreach (ParagraphItem paraItem in paragraph.ChildEntities)
        {
            //Gets the image from its title and modifies its width and height
            if (paraItem is WPicture)
            {
                WPicture picture = paraItem as WPicture;
                if (picture.Title == "Bookmark")
                {
                    picture.Width = 150;
                    picture.Height = 100;
                }
            }
        }
    }
}

```

```

}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Working with lists

Lists can organize and format the contents of a document in hierarchical way. There are nine levels in the list, starting from level 0 to level 8. DocIO supports both built-in list styles and custom list styles. The following are the types of list supported in DocIO:

- Numbered list
- Bulleted list

The following code example explains how to create a simple bulleted list.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefBulletStyle();
//Adds text to the paragraph
paragraph.AppendText("List item 1");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Applies default numbered list style
paragraph.ListFormat.ApplyDefBulletStyle()
'Adds text to the paragraph
paragraph.AppendText("List item 1")
'Continues the list defined
paragraph.ListFormat.ContinueListNumbering()
'Adds second paragraph
paragraph = section.AddParagraph()
paragraph.AppendText("List item 2")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Adds new paragraph
paragraph = section.AddParagraph()
paragraph.AppendText("List item 3")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefBulletStyle();
//Adds text to the paragraph
paragraph.AppendText("List item 1");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();

```



```
//Saves the stream as Word file in local machine  
Save(stream, "Result.docx");  
//Refer to the following link to save Word document in UWP platform  
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-  
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document  
WordDocument document = new WordDocument();  
//Adds new section to the document  
IWSection section = document.AddSection();  
//Adds new paragraph to the section  
IWParagraph paragraph = section.AddParagraph();  
//Applies default numbered list style  
paragraph.ListFormat.ApplyDefBulletStyle();  
//Adds text to the paragraph  
paragraph.AppendText("List item 1");  
//Continues the list defined  
paragraph.ListFormat.ContinueListNumbering();  
//Adds second paragraph  
paragraph = section.AddParagraph();  
paragraph.AppendText("List item 2");  
//Continues last defined list  
paragraph.ListFormat.ContinueListNumbering();  
//Adds new paragraph  
paragraph = section.AddParagraph();  
paragraph.AppendText("List item 3");  
//Continues last defined list  
paragraph.ListFormat.ContinueListNumbering();  
//Saves and closes the Word document instance  
MemoryStream stream = new MemoryStream();  
//Saves the Word document to MemoryStream  
document.Save(stream, FormatType.Docx);  
document.Close();  
stream.Position = 0;  
//Download Word document in the browser  
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document  
WordDocument document = new WordDocument();  
//Adds new section to the document  
IWSection section = document.AddSection();  
//Adds new paragraph to the section  
IWParagraph paragraph = section.AddParagraph();  
//Applies default numbered list style  
paragraph.ListFormat.ApplyDefBulletStyle();  
//Adds text to the paragraph  
paragraph.AppendText("List item 1");  
//Continues the list defined  
paragraph.ListFormat.ContinueListNumbering();  
//Adds second paragraph  
paragraph = section.AddParagraph();  
paragraph.AppendText("List item 2");
```

```

//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example explains how to create a simple numbered list.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle();
//Adds text to the paragraph
paragraph.AppendText("List item 1");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document

```

```

Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWPParagraph = section.AddParagraph()
'Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle()
'Adds text to the paragraph
paragraph.AppendText("List item 1")
'Continues the list defined
paragraph.ListFormat.ContinueListNumbering()
'Adds second paragraph
paragraph = section.AddParagraph()
paragraph.AppendText("List item 2")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Adds new paragraph
paragraph = section.AddParagraph()
paragraph.AppendText("List item 3")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWPParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle();
//Adds text to the paragraph
paragraph.AppendText("List item 1");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform

```

```
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle();
//Adds text to the paragraph
paragraph.AppendText("List item 1");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle();
//Adds text to the paragraph
paragraph.AppendText("List item 1");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Adds new paragraph
```

```

paragraph = section.AddParagraph();
paragraph.AppendText("List item 3");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example explains how to create a multilevel bulleted list.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefBulletStyle();
//Adds text to the paragraph
paragraph.AppendText("List item 1 - Level 0");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2 - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3 - Level 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()

```

```

'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Applies default numbered list style
paragraph.ListFormat.ApplyDefBulletStyle()
'Adds text to the paragraph
paragraph.AppendText("List item 1 - Level 0")
'Continues the list defined
paragraph.ListFormat.ContinueListNumbering()
'Adds second paragraph
paragraph = section.AddParagraph()
paragraph.AppendText("List item 2 - Level 1")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel()
'Adds new paragraph
paragraph = section.AddParagraph()
paragraph.AppendText("List item 3 - Level 2")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel()
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefBulletStyle();
//Adds text to the paragraph
paragraph.AppendText("List item 1 - Level 0");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2 - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3 - Level 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent

```

```

paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefBulletStyle();
//Adds text to the paragraph
paragraph.AppendText("List item 1 - Level 0");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2 - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3 - Level 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section

```

```

IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefBulletStyle();
//Adds text to the paragraph
paragraph.AppendText("List item 1 - Level 0");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2 - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3 - Level 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example explains how to create multilevel numbered list.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle();
//Adds text to the paragraph
paragraph.AppendText("List item 1 - Level 0");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2 - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent

```



```

paragraph.ListFormat.IncreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3 - Level 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle()
'Adds text to the paragraph
paragraph.AppendText("List item 1 - Level 0")
'Continues the list defined
paragraph.ListFormat.ContinueListNumbering()
'Adds second paragraph
paragraph = section.AddParagraph()
paragraph.AppendText("List item 2 - Level 1")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel()
'Adds new paragraph
paragraph = section.AddParagraph()
paragraph.AppendText("List item 3 - Level 2")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel()
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle();

```

```

//Adds text to the paragraph
paragraph.AppendText("List item 1 - Level 0");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2 - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3 - Level 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle();
//Adds text to the paragraph
paragraph.AppendText("List item 1 - Level 0");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2 - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3 - Level 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent

```

```

paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle();
//Adds text to the paragraph
paragraph.AppendText("List item 1 - Level 0");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 2 - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("List item 3 - Level 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The list levels can be incremented or decremented by using the `IncreaseIndentLevel` and `DecreaseIndentLevel` methods respectively. The following code example explains how to increase or decrease the list indent levels.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle();
//Adds text to the paragraph
paragraph.AppendText("Multilevel numbered list - Level 0");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("Multilevel numbered list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("Multilevel numbered list - Level 0");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.DecreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("Multilevel numbered list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle()
'Adds text to the paragraph
paragraph.AppendText("Multilevel numbered list - Level 0")
'Continues the list defined
paragraph.ListFormat.ContinueListNumbering()
'Adds second paragraph
paragraph = section.AddParagraph()

```

```

paragraph.AppendText("Multilevel numbered list - Level 1")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel()
'Adds new paragraph
paragraph = section.AddParagraph()
paragraph.AppendText("Multilevel numbered list - Level 0")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Increases the level indent
paragraph.ListFormat.DecreaseIndentLevel()
'Adds new paragraph
paragraph = section.AddParagraph()
paragraph.AppendText("Multilevel numbered list - Level 1")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel()
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle();
//Adds text to the paragraph
paragraph.AppendText("Multilevel numbered list - Level 0");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("Multilevel numbered list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("Multilevel numbered list - Level 0");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.DecreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("Multilevel numbered list - Level 1");
//Continues last defined list

```

```

paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle();
//Adds text to the paragraph
paragraph.AppendText("Multilevel numbered list - Level 0");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("Multilevel numbered list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("Multilevel numbered list - Level 0");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.DecreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("Multilevel numbered list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser

```

```
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Applies default numbered list style
paragraph.ListFormat.ApplyDefNumberedStyle();
//Adds text to the paragraph
paragraph.AppendText("Multilevel numbered list - Level 0");
//Continues the list defined
paragraph.ListFormat.ContinueListNumbering();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("Multilevel numbered list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("Multilevel numbered list - Level 0");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.DecreaseIndentLevel();
//Adds new paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("Multilevel numbered list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

The following code example explains how to create user defined list styles.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
```

```

//Adds new section to the document
IWSection section = document.AddSection();
//Adds new list style to the document
ListStyle listStyle = document.AddListStyle(ListType.Numbered,
"UserDefinedList");
WListLevel levelOne = listStyle.Levels[0];
//Defines the follow character, prefix, suffix, start index for level 0
levelOne.FollowCharacter = FollowCharacterType.Tab;
levelOne.NumberPrefix = "(";
levelOne.NumberSuffix = ")";
levelOne.PatternType = ListPatternType.LowRoman;
levelOne.StartAt = 1;
levelOne.TabSpaceAfter = 5;
levelOne.NumberAlignment = ListNumberAlignment.Center;
WListLevel levelTwo = listStyle.Levels[1];
//Defines the follow character, suffix, pattern, start index for level 1
levelTwo.FollowCharacter = FollowCharacterType.Tab;
levelTwo.NumberSuffix = "}";
levelTwo.PatternType = ListPatternType.LowLetter;
levelTwo.StartAt = 2;
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds text to the paragraph
paragraph.AppendText("User defined list - Level 0");
//Applies default numbered list style
paragraph.ListFormat.ApplyStyle("UserDefinedList");
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("User defined list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new list style to the document
Dim listStyle As ListStyle = document.AddListStyle(ListType.Numbered,
"UserDefinedList")
Dim levelOne As WListLevel = listStyle.Levels(0)
'Defines the follow character, prefix, suffix, start index for level 0
levelOne.FollowCharacter = FollowCharacterType.Tab
levelOne.NumberPrefix = "("
levelOne.NumberSuffix = ")"
levelOne.PatternType = ListPatternType.LowRoman
levelOne.StartAt = 1
levelOne.TabSpaceAfter = 5
levelOne.NumberAlignment = ListNumberAlignment.Center

```



```

Dim levelTwo As WListLevel = listStyle.Levels(1)
'Defines the follow character, suffix, pattern, start index for level 1
levelTwo.FollowCharacter = FollowCharacterType.Tab
levelTwo.NumberSuffix = "}"
levelTwo.PatternType = ListPatternType.LowLetter
levelTwo.StartAt = 2
'Adds new paragraph to the section
Dim paragraph As IWPParagraph = section.AddParagraph()
'Adds text to the paragraph
paragraph.AppendText("User defined list - Level 0")
'Applies default numbered list style
paragraph.ListFormat.ApplyStyle("UserDefinedList")
'Adds second paragraph
paragraph = section.AddParagraph()
paragraph.AppendText("User defined list - Level 1")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel()
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new list style to the document
ListStyle listStyle = document.AddListStyle(ListType.Numbered,
"UserDefinedList");
WListLevel levelOne = listStyle.Levels[0];
//Defines the follow character, prefix, suffix, start index for level 0
levelOne.FollowCharacter = FollowCharacterType.Tab;
levelOne.NumberPrefix = "(";
levelOne.NumberSuffix = ")";
levelOne.PatternType = ListPatternType.LowRoman;
levelOne.StartAt = 1;
levelOne.TabSpaceAfter = 5;
levelOne.NumberAlignment = ListNumberAlignment.Center;
WListLevel levelTwo = listStyle.Levels[1];
//Defines the follow character, suffix, pattern, start index for level 1
levelTwo.FollowCharacter = FollowCharacterType.Tab;
levelTwo.NumberSuffix = "}";
levelTwo.PatternType = ListPatternType.LowLetter;
levelTwo.StartAt = 2;
//Adds new paragraph to the section
IWPParagraph paragraph = section.AddParagraph();
//Adds text to the paragraph
paragraph.AppendText("User defined list - Level 0");
//Applies default numbered list style
paragraph.ListFormat.ApplyStyle("UserDefinedList");
//Adds second paragraph
paragraph = section.AddParagraph();

```

```

paragraph.AppendText("User defined list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new list style to the document
ListStyle listStyle = document.AddListStyle(ListType.Numbered,
"UserDefinedList");
WListLevel levelOne = listStyle.Levels[0];
//Defines the follow character, prefix, suffix, start index for level 0
levelOne.FollowCharacter = FollowCharacterType.Tab;
levelOne.NumberPrefix = "(";
levelOne.NumberSuffix = ")";
levelOne.PatternType = ListPatternType.LowRoman;
levelOne.StartAt = 1;
levelOne.TabSpaceAfter = 5;
levelOne.NumberAlignment = ListNumberAlignment.Center;
WListLevel levelTwo = listStyle.Levels[1];
//Defines the follow character, suffix, pattern, start index for level 1
levelTwo.FollowCharacter = FollowCharacterType.Tab;
levelTwo.NumberSuffix = "}";
levelTwo.PatternType = ListPatternType.LowLetter;
levelTwo.StartAt = 2;
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds text to the paragraph
paragraph.AppendText("User defined list - Level 0");
//Applies default numbered list style
paragraph.ListFormat.ApplyStyle("UserDefinedList");
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("User defined list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream

```

```
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new list style to the document
ListStyle listStyle = document.AddListStyle(ListType.Numbered,
"UserDefinedList");
WListLevel levelOne = listStyle.Levels[0];
//Defines the follow character, prefix, suffix, start index for level 0
levelOne.FollowCharacter = FollowCharacterType.Tab;
levelOne.NumberPrefix = "(";
levelOne.NumberSuffix = ")";
levelOne.PatternType = ListPatternType.LowRoman;
levelOne.StartAt = 1;
levelOne.TabSpaceAfter = 5;
levelOne.NumberAlignment = ListNumberAlignment.Center;
WListLevel levelTwo = listStyle.Levels[1];
//Defines the follow character, suffix, pattern, start index for level 1
levelTwo.FollowCharacter = FollowCharacterType.Tab;
levelTwo.NumberSuffix = "}";
levelTwo.PatternType = ListPatternType.LowLetter;
levelTwo.StartAt = 2;
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds text to the paragraph
paragraph.AppendText("User defined list - Level 0");
//Applies default numbered list style
paragraph.ListFormat.ApplyStyle("UserDefinedList");
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("User defined list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

The following code example explains how to create numbered list with prefix from previous level.

Note: The `NumberPrefix` value for the numbered list should meet the syntax "`\u000N`" to update the previous list level value as prefix to the current list level. For example, it should be represented as ("`\u0000.`" or "`\u0000.\u0001.`").

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new list style to the document
ListStyle listStyle = document.AddListStyle(ListType.Numbered,
"UserDefinedList");
WListLevel levelOne = listStyle.Levels[0];
//Defines the follow character, prefix from previous level, start index for
level 0
levelOne.FollowCharacter = FollowCharacterType.Nothing;
levelOne.PatternType = ListPatternType.Arabic;
levelOne.StartAt = 1;
WListLevel levelTwo = listStyle.Levels[1];
//Defines the follow character, prefix from previous level, pattern, start
index for level 1
levelTwo.FollowCharacter = FollowCharacterType.Nothing;
levelTwo.NumberPrefix = "\u0000.";
levelTwo.PatternType = ListPatternType.Arabic;
levelTwo.StartAt = 1;
WListLevel levelThree = listStyle.Levels[2];
//Defines the follow character, prefix from previous level, pattern, start
index for level 1
levelThree.FollowCharacter = FollowCharacterType.Nothing;
levelThree.NumberPrefix = "\u0000.\u0001.";
levelThree.PatternType = ListPatternType.Arabic;
levelThree.StartAt = 1;
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds text to the paragraph
paragraph.AppendText("User defined list - Level 0");
//Applies default numbered list style
paragraph.ListFormat.ApplyStyle("UserDefinedList");
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("User defined list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("User defined list - Level 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
```

```
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new list style to the document
Dim listStyle As ListStyle = document.AddListStyle(ListType.Numbered,
"UserDefinedList")
Dim levelOne As WListLevel = listStyle.Levels(0)
'Defines the follow character, prefix from previous level, start index for
level 0
levelOne.FollowCharacter = FollowCharacterType.[Nothing]
levelOne.PatternType = ListPatternType.Arabic
levelOne.StartAt = 1
Dim levelTwo As WListLevel = listStyle.Levels(1)
'Defines the follow character, prefix from previous level, pattern, start
index for level 1
levelTwo.FollowCharacter = FollowCharacterType.[Nothing]
levelTwo.NumberPrefix = vbNullChar & "."
levelTwo.PatternType = ListPatternType.Arabic
levelTwo.StartAt = 1
Dim levelThree As WListLevel = listStyle.Levels(2)
'Defines the follow character, prefix from previous level, pattern, start
index for level 1
levelThree.FollowCharacter = FollowCharacterType.[Nothing]
levelThree.NumberPrefix = vbNullChar & "." & ChrW(1) & "."
levelThree.PatternType = ListPatternType.Arabic
levelThree.StartAt = 1
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Adds text to the paragraph
paragraph.AppendText("User defined list - Level 0")
'Applies default numbered list style
paragraph.ListFormat.ApplyStyle("UserDefinedList")
'Adds second paragraph
paragraph = section.AddParagraph()
paragraph.AppendText("User defined list - Level 1")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel()
'Adds second paragraph
paragraph = section.AddParagraph()
paragraph.AppendText("User defined list - Level 2")
'Continues last defined list
paragraph.ListFormat.ContinueListNumbering()
'Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel()
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new list style to the document
ListStyle listStyle = document.AddListStyle(ListType.Numbered,
"UserDefinedList");
WListLevel levelOne = listStyle.Levels[0];
//Defines the follow character, prefix from previous level, start index for level 0
levelOne.FollowCharacter = FollowCharacterType.Nothing;
levelOne.PatternType = ListPatternType.Arabic;
levelOne.StartAt = 1;
WListLevel levelTwo = listStyle.Levels[1];
//Defines the follow character, prefix from previous level, pattern, start index for level 1
levelTwo.FollowCharacter = FollowCharacterType.Nothing;
levelTwo.NumberPrefix = "\u0000.";
levelTwo.PatternType = ListPatternType.Arabic;
levelTwo.StartAt = 1;
WListLevel levelThree = listStyle.Levels[2];
//Defines the follow character, prefix from previous level, pattern, start index for level 1
levelThree.FollowCharacter = FollowCharacterType.Nothing;
levelThree.NumberPrefix = "\u0000.\u0001.";
levelThree.PatternType = ListPatternType.Arabic;
levelThree.StartAt = 1;
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds text to the paragraph
paragraph.AppendText("User defined list - Level 0");
//Applies default numbered list style
paragraph.ListFormat.ApplyStyle("UserDefinedList");
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("User defined list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("User defined list - Level 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine

```

```
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new list style to the document
ListStyle listStyle = document.AddListStyle(ListType.Numbered,
"UserDefinedList");
WListLevel levelOne = listStyle.Levels[0];
//Defines the follow character, prefix from previous level, start index for
level 0
levelOne.FollowCharacter = FollowCharacterType.Nothing;
levelOne.PatternType = ListPatternType.Arabic;
levelOne.StartAt = 1;
WListLevel levelTwo = listStyle.Levels[1];
//Defines the follow character, prefix from previous level, pattern, start
index for level 1
levelTwo.FollowCharacter = FollowCharacterType.Nothing;
levelTwo.NumberPrefix = "\u0000.";
levelTwo.PatternType = ListPatternType.Arabic;
levelTwo.StartAt = 1;
WListLevel levelThree = listStyle.Levels[2];
//Defines the follow character, prefix from previous level, pattern, start
index for level 1
levelThree.FollowCharacter = FollowCharacterType.Nothing;
levelThree.NumberPrefix = "\u0000.\u0001.";
levelThree.PatternType = ListPatternType.Arabic;
levelThree.StartAt = 1;
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds text to the paragraph
paragraph.AppendText("User defined list - Level 0");
//Applies default numbered list style
paragraph.ListFormat.ApplyStyle("UserDefinedList");
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("User defined list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("User defined list - Level 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
```

```
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new list style to the document
ListStyle listStyle = document.AddListStyle(ListType.Numbered,
"UserDefinedList");
WListLevel levelOne = listStyle.Levels[0];
//Defines the follow character, prefix from previous level, start index for
level 0
levelOne.FollowCharacter = FollowCharacterType.Nothing;
levelOne.PatternType = ListPatternType.Arabic;
levelOne.StartAt = 1;
WListLevel levelTwo = listStyle.Levels[1];
//Defines the follow character, prefix from previous level, pattern, start
index for level 1
levelTwo.FollowCharacter = FollowCharacterType.Nothing;
levelTwo.NumberPrefix = "\u0000.";
levelTwo.PatternType = ListPatternType.Arabic;
levelTwo.StartAt = 1;
WListLevel levelThree = listStyle.Levels[2];
//Defines the follow character, prefix from previous level, pattern, start
index for level 1
levelThree.FollowCharacter = FollowCharacterType.Nothing;
levelThree.NumberPrefix = "\u0000.\u0001.";
levelThree.PatternType = ListPatternType.Arabic;
levelThree.StartAt = 1;
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds text to the paragraph
paragraph.AppendText("User defined list - Level 0");
//Applies default numbered list style
paragraph.ListFormat.ApplyStyle("UserDefinedList");
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("User defined list - Level 1");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
//Adds second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText("User defined list - Level 2");
//Continues last defined list
paragraph.ListFormat.ContinueListNumbering();
//Increases the level indent
paragraph.ListFormat.IncreaseIndentLevel();
```



```
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Get list value

You can get the string that represents the appearance of **list value of the paragraph** in the Word document using the `ListString` API.

This API holds the static string of the list value recently calculated while saving the document as Text. It is not updated automatically for each modification done in the Word document. Hence, you should either invoke the `GetText()` method of `WordDocument` or save the Word document as Text to get the actual list value from this API.

The following example shows how to **get a string that represents the appearance of list value of the paragraph**.

C#

```
//Loads an existing Word document
WordDocument document = new WordDocument("Template.docx");
//Gets the document text
document.GetText();
//Gets the string that represents the appearance of list value of the
paragraph
String listString = document.LastParagraph.ListString;
//Saves and closes the WordDocument instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Loads an existing Word document
Dim document As WordDocument = New WordDocument("Template.docx")
' Gets the document text
document.GetText()
'Gets the string that represents the appearance of list value of the
paragraph
Dim listString As String = document.LastParagraph.ListString
'Saves and closes the WordDocument instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Creates an instance of a WordDocument
```

```

WordDocument document = new WordDocument();
//Loads an existing Word document
document.Open(assembly.GetManifestResourceStream("Sample.Assets.Template.docx"), FormatType.Docx);
//Gets the document text
document.GetText();
//Gets the string that represents the appearance of list value of the paragraph
String listString = document.LastParagraph.ListString;
// Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Closes the Word document
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Refer to the following link to save Word document in UWP platform.
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads an existing Word document
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Gets the document text
document.GetText();
//Gets the string that represents the appearance of list value of the paragraph
String listString = document.LastParagraph.ListString;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Loads an existing Word document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx"), FormatType.Docx);
//Gets the document text
document.GetText();
//Gets the string that represents the appearance of list value of the paragraph
String listString = document.LastParagraph.ListString;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the Word document

```

```
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform.
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Note: For a picture bulleted list, the `ListString` API is not valid and it will return an empty string.

Working with hyperlinks

Hyperlink is a reference to data that can link to external contents like images, files, webpage, and more. In Word document, a hyperlink may target to any one of the following sources:

- Webpage: Represents the web content.
- File: Represents the file in some location.
- Email: Represents an Email.
- Bookmark: Represents the bookmarks in the document.

Hyperlinks have two parts: the address and display content.

The following code example explains how to insert a web link.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Web Hyperlink: ");
paragraph = section.AddParagraph();
//Appends web hyperlink to the paragraph
IWField field = paragraph.AppendHyperlink("http://www.syncfusion.com",
"Syncfusion", HyperlinkType.WebLink);
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
paragraph.AppendText("Web Hyperlink: ")
paragraph = section.AddParagraph()
'Appends web hyperlink to the paragraph
```

```

Dim field As IField =
paragraph.AppendHyperlink("http://www.syncfusion.com", "Syncfusion",
HyperlinkType.WebLink)
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Web Hyperlink: ");
paragraph = section.AddParagraph();
//Appends web hyperlink to the paragraph
IField field = paragraph.AppendHyperlink("http://www.syncfusion.com",
"Syncfusion", HyperlinkType.WebLink);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Web Hyperlink: ");
paragraph = section.AddParagraph();
//Appends web hyperlink to the paragraph
IField field = paragraph.AppendHyperlink("http://www.syncfusion.com",
"Syncfusion", HyperlinkType.WebLink);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Web Hyperlink: ");
paragraph = section.AddParagraph();
//Appends web hyperlink to the paragraph
IWField field = paragraph.AppendHyperlink("http://www.syncfusion.com",
"Syncfusion", HyperlinkType.WebLink);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example illustrates how to add an email link.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Email hyperlink: ");
paragraph = section.AddParagraph();
//Appends Email hyperlink to the paragraph
paragraph.AppendHyperlink("mailto:sales@syncfusion.com", "Sales" ,
HyperlinkType.EmailLink);
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
paragraph.AppendText("Email hyperlink: ")
paragraph = section.AddParagraph()
'Appends Email hyperlink to the paragraph

```

```

paragraph.AppendHyperlink("mailto:sales@syncfusion.com", "Sales" ,
HyperlinkType.EmailLink)
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Email hyperlink: ");
paragraph = section.AddParagraph();
//Appends Email hyperlink to the paragraph
paragraph.AppendHyperlink("mailto:sales@syncfusion.com", "Sales",
HyperlinkType.EmailLink);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Email hyperlink: ");
paragraph = section.AddParagraph();
//Appends Email hyperlink to the paragraph
paragraph.AppendHyperlink("mailto:sales@syncfusion.com", "Sales",
HyperlinkType.EmailLink);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Email hyperlink: ");
paragraph = section.AddParagraph();
//Appends Email hyperlink to the paragraph
paragraph.AppendHyperlink("mailto:sales@syncfusion.com", "Sales",
HyperlinkType.EmailLink);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example explains how to add a file hyperlink.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("File Hyperlinks: ");
paragraph = section.AddParagraph();
//Appends hyperlink field to the paragraph
paragraph.AppendHyperlink(@"Template.docx", "File", HyperlinkType.FileLink);
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
paragraph.AppendText("File Hyperlinks: ")
paragraph = section.AddParagraph()
'Appends hyperlink field to the paragraph
paragraph.AppendHyperlink("Template.docx", "File", HyperlinkType.FileLink)
'Saves the Word document

```

```
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("File Hyperlinks: ");
paragraph = section.AddParagraph();
//Appends hyperlink field to the paragraph
paragraph.AppendHyperlink(@"Template.docx", "File", HyperlinkType.FileLink);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("File Hyperlinks: ");
paragraph = section.AddParagraph();
//Appends hyperlink field to the paragraph
paragraph.AppendHyperlink(@"Template.docx", "File", HyperlinkType.FileLink);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
```



```

IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("File Hyperlinks: ");
paragraph = section.AddParagraph();
//Appends hyperlink field to the paragraph
paragraph.AppendHyperlink(@"Template.docx", "File", HyperlinkType.FileLink);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example explains how to add a bookmark hyperlink.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Creates new Bookmark
paragraph.AppendBookmarkStart("Introduction");
paragraph.AppendText("Hyperlink");
paragraph.AppendBookmarkEnd("Introduction");
paragraph.AppendText("\nA hyperlink is a reference or navigation element in
a document to another section of the same document or to another document
that may be on or part of a (different) domain.");
paragraph = section.AddParagraph();
paragraph.AppendText("Bookmark Hyperlink: ");
paragraph = section.AddParagraph();
//Appends Bookmark hyperlink to the paragraph
paragraph.AppendHyperlink("Introduction", "Bookmark",
HyperlinkType.Bookmark);
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Creates new Bookmark
paragraph.AppendBookmarkStart("Introduction")

```

```

paragraph.AppendText("Hyperlink")
paragraph.AppendBookmarkEnd("Introduction")
paragraph.AppendText(vbLf & "A hyperlink is a reference or navigation
element in a document to another section of the same document or to another
document that may be on or part of a (different) domain.")
paragraph = section.AddParagraph()
paragraph.AppendText("Bookmark Hyperlink: ")
paragraph = section.AddParagraph()
'Appends Bookmark hyperlink to the paragraph
paragraph.AppendHyperlink("Introduction", "Bookmark",
HyperlinkType.Bookmark)
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Creates new Bookmark
paragraph.AppendBookmarkStart("Introduction");
paragraph.AppendText("Hyperlink");
paragraph.AppendBookmarkEnd("Introduction");
paragraph.AppendText("\nA hyperlink is a reference or navigation element in
a document to another section of the same document or to another document
that may be on or part of a (different) domain.");
paragraph = section.AddParagraph();
paragraph.AppendText("Bookmark Hyperlink: ");
paragraph = section.AddParagraph();
//Appends Bookmark hyperlink to the paragraph
paragraph.AppendHyperlink("Introduction", "Bookmark",
HyperlinkType.Bookmark);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();

```

```
//Creates new Bookmark
paragraph.AppendBookmarkStart("Introduction");
paragraph.AppendText("Hyperlink");
paragraph.AppendBookmarkEnd("Introduction");
paragraph.AppendText("\nA hyperlink is a reference or navigation element in
a document to another section of the same document or to another document
that may be on or part of a (different) domain.");
paragraph = section.AddParagraph();
paragraph.AppendText("Bookmark Hyperlink: ");
paragraph = section.AddParagraph();
//Appends Bookmark hyperlink to the paragraph
paragraph.AppendHyperlink("Introduction", "Bookmark",
HyperlinkType.Bookmark);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Creates new Bookmark
paragraph.AppendBookmarkStart("Introduction");
paragraph.AppendText("Hyperlink");
paragraph.AppendBookmarkEnd("Introduction");
paragraph.AppendText("\nA hyperlink is a reference or navigation element in
a document to another section of the same document or to another document
that may be on or part of a (different) domain.");
paragraph = section.AddParagraph();
paragraph.AppendText("Bookmark Hyperlink: ");
paragraph = section.AddParagraph();
//Appends Bookmark hyperlink to the paragraph
paragraph.AppendHyperlink("Introduction", "Bookmark",
HyperlinkType.Bookmark);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

The display content for the Hyperlinks can also be an image that may redirect to some other contents.

The following code example explains how to add image hyperlink.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Image Hyperlink");
paragraph = section.AddParagraph();
//Creates a new image instance and load image
WPicture picture = new WPicture(document);
picture.LoadImage(Image.FromFile("Image.png"));
//Appends new image hyperlink to the paragraph
paragraph.AppendHyperlink("http://www.syncfusion.com", picture,
HyperlinkType.WebLink);
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
paragraph.AppendText("Image Hyperlink")
paragraph = section.AddParagraph()
'Creates a new image instance and load image
Dim picture As New WPicture(document)
picture.LoadImage(Image.FromFile("Image.png"))
'Appends new image hyperlink to the paragraph
paragraph.AppendHyperlink("http://www.syncfusion.com", picture,
HyperlinkType.WebLink)
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Image Hyperlink");
paragraph = section.AddParagraph();
```

```

//Creates a new image instance and load image
WPicture picture = new WPicture(document);
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream imageStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Dummy-
Images.jpg");
picture.LoadImage(imageStream);
//Appends new image hyperlink to the paragraph
paragraph.AppendHyperlink("http://www.syncfusion.com", picture,
HyperlinkType.WebLink);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Image Hyperlink");
paragraph = section.AddParagraph();
//Creates a new image instance and load image
WPicture picture = new WPicture(document);
FileStream imageStream = new FileStream(@"Mountain-200.jpg", FileMode.Open,
FileAccess.ReadWrite);
picture.LoadImage(imageStream);
//Appends new image hyperlink to the paragraph
paragraph.AppendHyperlink("http://www.syncfusion.com", picture,
HyperlinkType.WebLink);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();

```

```

paragraph.AppendText("Image Hyperlink");
paragraph = section.AddParagraph();
//Creates a new image instance and load image
WPicture picture = new WPicture(document);
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream imageStream =
assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Dummy-
Images.jpg");
picture.LoadImage(imageStream);
//Appends new image hyperlink to the paragraph
paragraph.AppendHyperlink("http://www.syncfusion.com", picture,
HyperlinkType.WebLink);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example explains how to modify the URL of an existing hyperlink.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Sample.docx", FormatType.Docx);
WParagraph paragraph = document.LastParagraph;
//Iterates through the paragraph items
foreach (ParagraphItem item in paragraph.ChildEntities)
{
    if (item is WField)
    {
        if ((item as WField).FieldType == FieldType.FieldHyperlink)
        {
            //Gets the hyperlink field
            Hyperlink link = new Hyperlink(item as WField);
            if (link.Type == HyperlinkType.WebLink)
            {
                //Modifies the url of the hyperlink
                link.Uri = "http://www.google.com";
                link.TextToDisplay = "Google";
                break;
            }
        }
    }
}
//Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Sample.docx", FormatType.Docx)
Dim paragraph As WParagraph = document.LastParagraph
'Iterates through the paragraph items
For Each item As ParagraphItem In paragraph.ChildEntities
If TypeOf item Is WField Then
If TryCast(item, WField).FieldType = FieldType.FieldHyperlink Then
'Gets the hyperlink field
Dim link As New Hyperlink(TryCast(item, WField))
If link.Type = HyperlinkType.WebLink Then
'Modifies the url of the hyperlink
link.Uri = "http://www.google.com"
link.TextToDisplay = "Google"
Exit For
End If
End If
End If
Next
'Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Sample.docx");
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatType.Docx);
WParagraph paragraph = document.LastParagraph;
//Iterates through the paragraph items
foreach (ParagraphItem item in paragraph.ChildEntities)
{
if (item is WField)
{
if ((item as WField).FieldType == FieldType.FieldHyperlink)
{
//Gets the hyperlink field
Hyperlink link = new Hyperlink(item as WField);
if (link.Type == HyperlinkType.WebLink)
{
//Modifies the url of the hyperlink
link.Uri = "http://www.google.com";
link.TextToDisplay = "Google";
break;
}
}
}
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();

```

```
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
FileStream fileStream = new FileStream(@"Sample.docx", FileMode.Open,
    FileAccess.ReadWrite);
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatType.Docx);
WParagraph paragraph = document.LastParagraph;
//Iterates through the paragraph items
foreach (ParagraphItem item in paragraph.ChildEntities)
{
    if (item is WField)
    {
        if ((item as WField).FieldType == FieldType.FieldHyperlink)
        {
            //Gets the hyperlink field
            Hyperlink link = new Hyperlink(item as WField);
            if (link.Type == HyperlinkType.WebLink)
            {
                //Modifies the url of the hyperlink
                link.Uri = "http://www.google.com";
                link.TextToDisplay = "Google";
                break;
            }
        }
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
    assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Sample.docx");
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatType.Docx);
WParagraph paragraph = document.LastParagraph;
//Iterates through the paragraph items
foreach (ParagraphItem item in paragraph.ChildEntities)
{
    if (item is WField)
    {
        if ((item as WField).FieldType == FieldType.FieldHyperlink)
        {
```



```

//Gets the hyperlink field
Hyperlink link = new Hyperlink(item as WField);
if (link.Type == HyperlinkType.WebLink)
{
    //Modifies the url of the hyperlink
    link.Uri = "http://www.google.com";
    link.TextToDisplay = "Google";
    break;
}
}
}
}

//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Working with symbols

Symbols are used to add contents such as currencies, numbers, punctuations, etc. DocIO represents symbols with **WSymbol** instance. Each symbol can be identified with their character codes.

The following code example explains how to add new symbol to the document.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Example of adding symbols to the paragraph: ");
//Inserts symbol with character code 100
paragraph.AppendSymbol(100);
//Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
paragraph.AppendText("Example of adding symbols to the paragraph: ")
'Inserts symbol with character code 100

```

```
paragraph.AppendSymbol(100)
'Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Example of adding symbols to the paragraph: ");
//Inserts symbol with character code 100
paragraph.AppendSymbol(100);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Example of adding symbols to the paragraph: ");
//Inserts symbol with character code 100
paragraph.AppendSymbol(100);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
```

```

paragraph.AppendText("Example of adding symbols to the paragraph: ");
//Inserts symbol with character code 100
paragraph.AppendSymbol(100);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example explains how to modify an existing symbol.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Sample.docx", FormatType.Docx);
//Gets the textbody content
WTextBody textbody = document.Sections[0].Body;
//Iterates through the paragraphs
foreach (WParagraph paragraph in textbody.Paragraphs)
{
    //Gets the symbol from the paragraph items
    foreach (ParagraphItem item in paragraph.ChildEntities)
    {
        if (item is WSymbol)
        {
            WSymbol symbol = item as WSymbol;
            if (symbol.CharacterCode == 100)
            {
                //Modifies the character code
                symbol.CharacterCode = 40;
                symbol.FontName = "Wingdings";
            }
        }
    }
}
//Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Sample.docx", FormatType.Docx)
'Gets the textbody content
Dim textbody As WTextBody = document.Sections(0).Body
'Iterates through the paragraphs
For Each paragraph As WParagraph In textbody.Paragraphs
    'Gets the symbol from the paragraph items
    For Each item As ParagraphItem In paragraph.ChildEntities

```

```

If TypeOf item Is WSymbol Then
Dim symbol As WSymbol = TryCast(item, WSymbol)
If symbol.CharacterCode = 100 Then
    'Modifies the character code
    symbol.CharacterCode = 40
    symbol.FontName = "Wingdings"
End If
End If
Next
Next
'Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Sample.docx");
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatType.Docx);
//Gets the textbody content
WTextBody textbody = document.Sections[0].Body;
//Iterates through the paragraphs
foreach (WParagraph paragraph in textbody.Paragraphs)
{
    //Gets the symbol from the paragraph items
    foreach (ParagraphItem item in paragraph.ChildEntities)
    {
        if (item is WSymbol)
        {
            WSymbol symbol = item as WSymbol;
            if (symbol.CharacterCode == 100)
            {
                //Modifies the character code
                symbol.CharacterCode = 40;
                symbol.FontName = "Wingdings";
            }
        }
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

FileStream fileStream = new FileStream(@"Sample1.docx", FileMode.Open,
FileAccess.ReadWrite);

```

```
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatType.Docx);
//Gets the textbody content
WTextBody textbody = document.Sections[0].Body;
//Iterates through the paragraphs
foreach (WParagraph paragraph in textbody.Paragraphs)
{
    //Gets the symbol from the paragraph items
    foreach (ParagraphItem item in paragraph.ChildEntities)
    {
        if (item is WSymbol)
        {
            WSymbol symbol = item as WSymbol;
            if (symbol.CharacterCode == 100)
            {
                //Modifies the character code
                symbol.CharacterCode = 40;
                symbol.FontName = "Wingdings";
            }
        }
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Sample1.docx");
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatType.Docx);
//Gets the textbody content
WTextBody textbody = document.Sections[0].Body;
//Iterates through the paragraphs
foreach (WParagraph paragraph in textbody.Paragraphs)
{
    //Gets the symbol from the paragraph items
    foreach (ParagraphItem item in paragraph.ChildEntities)
    {
        if (item is WSymbol)
        {
            WSymbol symbol = item as WSymbol;
            if (symbol.CharacterCode == 100)
            {
                //Modifies the character code
                symbol.CharacterCode = 40;
                symbol.FontName = "Wingdings";
            }
        }
    }
}
```

```

}
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Appending breaks

Breaks allows the document contents to split into multiple parts to customize the appearance of the contents. The following are the types of breaks supported in the DocIO:

- Page break: Starts the content in the next page.
- Line break: Starts the content in new line.
- Column break: Starts the content in the next column.

The following code example explains how various types of breaks can be appended to the paragraphs.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Before line break");
//Adds line break to the paragraph
paragraph.AppendBreak(BreakType.LineBreak);
paragraph.AppendText("After line break");
IWParagraph pageBreakPara = section.AddParagraph();
pageBreakPara.AppendText("Before page break");
//Adds page break to the paragraph
pageBreakPara.AppendBreak(BreakType.PageBreak);
pageBreakPara.AppendText("After page break");
IWSection secondSection = document.AddSection();
//Adds columns to the section
secondSection.AddColumn(100, 2);
secondSection.AddColumn(100, 2);
IWParagraph columnBreakPara = secondSection.AddParagraph();
columnBreakPara.AppendText("Before column break");
//Adds column break to the paragraph
columnBreakPara.AppendBreak(BreakType.ColumnBreak);
columnBreakPara.AppendText("After column break");
//Saves and closes the document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As ISection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IParagraph = section.AddParagraph()
paragraph.AppendText("Before line break")
'Adds line break to the paragraph
paragraph.AppendBreak(BreakType.LineBreak)
paragraph.AppendText("After line break")
Dim pageBreakPara As IParagraph = section.AddParagraph()
pageBreakPara.AppendText("Before page break")
'Adds page break to the paragraph
pageBreakPara.AppendBreak(BreakType.PageBreak)
pageBreakPara.AppendText("After page break")
Dim secondSection As ISection = document.AddSection()
'Adds columns to the section
secondSection.AddColumn(100, 2)
secondSection.AddColumn(100, 2)
Dim columnBreakPara As IParagraph = secondSection.AddParagraph()
columnBreakPara.AppendText("Before column break")
'Adds column break to the paragraph
columnBreakPara.AppendBreak(BreakType.ColumnBreak)
columnBreakPara.AppendText("After column break")
'Saves and closes the document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
ISection section = document.AddSection();
//Adds new paragraph to the section
IParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Before line break");
//Adds line break to the paragraph
paragraph.AppendBreak(BreakType.LineBreak);
paragraph.AppendText("After line break");
IParagraph pageBreakPara = section.AddParagraph();
pageBreakPara.AppendText("Before page break");
//Adds page break to the paragraph
pageBreakPara.AppendBreak(BreakType.PageBreak);
pageBreakPara.AppendText("After page break");
ISection secondSection = document.AddSection();
//Adds columns to the section
secondSection.AddColumn(100, 2);
secondSection.AddColumn(100, 2);
IParagraph columnBreakPara = secondSection.AddParagraph();
columnBreakPara.AppendText("Before column break");
//Adds column break to the paragraph
columnBreakPara.AppendBreak(BreakType.ColumnBreak);
columnBreakPara.AppendText("After column break");

```

```
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Before line break");
//Adds line break to the paragraph
paragraph.AppendBreak(BreakType.LineBreak);
paragraph.AppendText("After line break");
IWParagraph pageBreakPara = section.AddParagraph();
pageBreakPara.AppendText("Before page break");
//Adds page break to the paragraph
pageBreakPara.AppendBreak(BreakType.PageBreak);
pageBreakPara.AppendText("After page break");
IWSection secondSection = document.AddSection();
//Adds columns to the section
secondSection.AddColumn(100, 2);
secondSection.AddColumn(100, 2);
IWParagraph columnBreakPara = secondSection.AddParagraph();
columnBreakPara.AppendText("Before column break");
//Adds column break to the paragraph
columnBreakPara.AppendBreak(BreakType.ColumnBreak);
columnBreakPara.AppendText("After column break");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Before line break");
//Adds line break to the paragraph
```



```

paragraph.AppendBreak(BreakType.LineBreak);
paragraph.AppendText("After line break");
IWParagraph pageBreakPara = section.AddParagraph();
pageBreakPara.AppendText("Before page break");
//Adds page break to the paragraph
pageBreakPara.AppendBreak(BreakType.PageBreak);
pageBreakPara.AppendText("After page break");
IWSection secondSection = document.AddSection();
//Adds columns to the section
secondSection.AddColumn(100, 2);
secondSection.AddColumn(100, 2);
IWParagraph columnBreakPara = secondSection.AddParagraph();
columnBreakPara.AppendText("Before column break");
//Adds column break to the paragraph
columnBreakPara.AppendBreak(BreakType.ColumnBreak);
columnBreakPara.AppendText("After column break");
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Appending OLE objects

OLE (Object Linking and Embedding) objects allow embedding and linking to documents and other objects. It allows the content of one program to be used in a Word document. The Objects can be inserted in the following two ways:

- Linked: The content is linked to the source file
- Embedded: The content is copied to the Word document and is not linked to the source file

You can create and manipulate the OLE Objects of both Linked and Embedded types in the Word document by using **WOleObject** instance.

The following code example explains how to add OLE objects to the document.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Opens the file to be embedded
FileStream stream = new FileStream("Book1.xlsx", FileMode.Open);
//Loads the picture instance with the image need to be displayed
WPicture picture = new WPicture(document);
picture.LoadImage(Image.FromFile("Image.png"));

```

```
//Appends the OLE object to the paragraph
WoleObject object = paragraph.AppendOleObject(stream, picture,
OleObjectType.ExcelWorksheet);
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Opens the file to be embedded
Dim stream As New FileStream("Book1.xlsx", FileMode.Open)
'Loads the picture instance with the image need to be displayed
Dim picture As New WPicture(document)
picture.LoadImage(Image.FromFile("Image.png"))
'Appends the OLE object to the paragraph
Dim object As WoleObject = paragraph.AppendOleObject(stream, picture,
OleObjectType.ExcelWorksheet)
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Opens the file to be embedded
Stream fileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Book1.xlsx");
//Loads the picture instance with the image need to be displayed
WPicture picture = new WPicture(document);
Stream imageStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Image.png");
picture.LoadImage(imageStream);
//Appends the OLE object to the paragraph
WoleObject oleObject = paragraph.AppendOleObject(fileStream, picture,
OleObjectType.ExcelWorksheet);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
```

*//Refer to the following link to save Word document in UWP platform
 //https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp*

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Opens the file to be embedded
FileStream fileStream = new FileStream("Book1.xlsx", FileMode.Open);
//Loads the picture instance with the image need to be displayed
WPicture picture = new WPicture(document);
FileStream imageStream = new FileStream(@"Image.png", FileMode.Open,
FileAccess.ReadWrite);
picture.LoadImage(imageStream);
//Appends the OLE object to the paragraph
WoleObject oleObject = paragraph.AppendOleObject(fileStream, picture,
OleObjectType.ExcelWorksheet);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Opens the file to be embedded
Stream fileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Book1.xlsx");
//Loads the picture instance with the image need to be displayed
WPicture picture = new WPicture(document);
Stream imageStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Image.png");
picture.LoadImage(imageStream);
//Appends the OLE object to the paragraph
WoleObject oleObject = paragraph.AppendOleObject(fileStream, picture,
OleObjectType.ExcelWorksheet);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
```

```
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Working with Text Box

Text box contains a group of textual and graphical contents. DocIO supports to create and manipulate the text box and its formatting by using the `WTextBox` instance.

The following code example explains how to add new text box to the paragraph.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends new textbox to the paragraph
IWTextBox textbox = paragraph.AppendTextBox(150, 75);
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
textboxParagraph.AppendText("Text inside text box");
textboxParagraph = textbox.TextBoxBody.AddParagraph();
//Adds new picture to textbox body
IWPicture picture =
textboxParagraph.AppendPicture(Image.FromFile(@"Image.png"));
picture.Height = 75;
picture.Width = 50;
//Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Appends new textbox to the paragraph
Dim textbox As IWTextBox = paragraph.AppendTextBox(150, 75)
'Adds new text to the textbox body
Dim textboxParagraph As IWParagraph = textbox.TextBoxBody.AddParagraph()
textboxParagraph.AppendText("Text inside text box")
textboxParagraph = textbox.TextBoxBody.AddParagraph()
'Adds new picture to textbox body
Dim picture As IWPicture =
textboxParagraph.AppendPicture(Image.FromFile("Image.png"))
picture.Height = 75
picture.Width = 50
```

```
'Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends new textbox to the paragraph
IWTextBox textbox = paragraph.AppendTextBox(150, 75);
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
textboxParagraph.AppendText("Text inside text box");
textboxParagraph = textbox.TextBoxBody.AddParagraph();
//Adds new picture to textbox body
Stream imageStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Dummy-
Images.jpg");
IWPicture picture = textboxParagraph.AppendPicture(imageStream);
picture.Height = 75;
picture.Width = 50;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends new textbox to the paragraph
IWTextBox textbox = paragraph.AppendTextBox(150, 75);
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
textboxParagraph.AppendText("Text inside text box");
textboxParagraph = textbox.TextBoxBody.AddParagraph();
//Adds new picture to textbox body
FileStream imageStream = new FileStream(@"Mountain-200.jpg", FileMode.Open,
FileAccess.ReadWrite);
IWPicture picture = textboxParagraph.AppendPicture(imageStream);
picture.Height = 75;
```

```

picture.Width = 50;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends new textbox to the paragraph
IWTextBox textbox = paragraph.AppendTextBox(150, 75);
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
textboxParagraph.AppendText("Text inside text box");
textboxParagraph = textbox.TextBoxBody.AddParagraph();
//Adds new picture to textbox body
Stream imageStream =
assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Dummy-
Images.jpg");
IWPicture picture = textboxParagraph.AppendPicture(imageStream);
picture.Height = 75;
picture.Width = 50;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Format and rotate text box

Text box has its own formatting such as outline color, fill effects, text direction, wrap formats, and more. You can also rotate the text box and apply flipping (horizontal and vertical) to it.

The following code example explains how to apply formatting and rotation for text box.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document

```

```

IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends new textbox to the paragraph
IWTextBox textbox = paragraph.AppendTextBox(150, 75);
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
textboxParagraph.AppendText("Text inside text box");
//Sets fill color and line width for textbox
textbox.TextBoxFormat.FillColor = Color.LightGreen;
textbox.TextBoxFormat.LineWidth = 2;
//Applies textbox text direction
textbox.TextBoxFormat.TextDirection =
Syncfusion.DocIO.DLS.TextDirection.VerticalTopToBottom;
//Sets text wrapping style
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical position
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Margin;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Sets top and bottom margin values
textbox.TextBoxFormat.InternalMargin.Bottom = 5f;
textbox.TextBoxFormat.InternalMargin.Top = 5f;
//Sets 90 degree rotation
textbox.TextBoxFormat.Rotation = 90;
//Sets horizontal flip
textbox.TextBoxFormat.FlipHorizontal = true;
//Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Appends new textbox to the paragraph
Dim textbox As IWTextBox = paragraph.AppendTextBox(150, 75)
'Adds new text to the textbox body
Dim textboxParagraph As IWParagraph = textbox.TextBoxBody.AddParagraph()
textboxParagraph.AppendText("Text inside text box")
'Sets fill color and line width for textbox
textbox.TextBoxFormat.FillColor = Color.LightGreen
textbox.TextBoxFormat.LineWidth = 2
'Applies textbox text direction
textbox.TextBoxFormat.TextDirection =
Syncfusion.DocIO.DLS.TextDirection.VerticalTopToBottom
'Sets text wrapping style
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText
'Sets horizontal and vertical position
textbox.TextBoxFormat.HorizontalPosition = 200

```

```

textbox.TextBoxFormat.VerticalPosition = 200
'Sets horizontal and vertical origin
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Margin
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page
'Sets top and bottom margin values
textbox.TextBoxFormat.InternalMargin.Bottom = 5.0F
textbox.TextBoxFormat.InternalMargin.Top = 5.0F
'Sets 90 degree rotation
textbox.TextBoxFormat.Rotation = 90
'Sets horizontal flip
textbox.TextBoxFormat.FlipHorizontal = true
'Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends new textbox to the paragraph
IWTextBox textbox = paragraph.AppendTextBox(150, 75);
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
textboxParagraph.AppendText("Text inside text box");
//Sets fill color and line width for textbox
textbox.TextBoxFormat.FillColor = Color.LightGreen;
textbox.TextBoxFormat.LineWidth = 2;
//Applies textbox text direction
textbox.TextBoxFormat.TextDirection =
Syncfusion.DocIO.DLS.TextDirection.VerticalTopToBottom;
//Sets text wrapping style
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical position
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Margin;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Sets top and bottom margin values
textbox.TextBoxFormat.InternalMargin.Bottom = 5f;
textbox.TextBoxFormat.InternalMargin.Top = 5f;
//Sets 90 degree rotation
textbox.TextBoxFormat.Rotation = 90;
//Sets horizontal flip
textbox.TextBoxFormat.FlipHorizontal = true;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");

```


*//Refer to the following link to save Word document in UWP platform
<https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp>*

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends new textbox to the paragraph
IWTextBox textbox = paragraph.AppendTextBox(150, 75);
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
textboxParagraph.AppendText("Text inside text box");
//Sets fill color and line width for textbox
textbox.TextBoxFormat.FillColor = Color.LightGreen;
textbox.TextBoxFormat.LineWidth = 2;
//Applies textbox text direction
textbox.TextBoxFormat.TextDirection =
Syncfusion.DocIO.DLS.TextDirection.VerticalTopToBottom;
//Sets text wrapping style
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical position
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Margin;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Sets top and bottom margin values
textbox.TextBoxFormat.InternalMargin.Bottom = 5f;
textbox.TextBoxFormat.InternalMargin.Top = 5f;
//Sets 90 degree rotation
textbox.TextBoxFormat.Rotation = 90;
//Sets horizontal flip
textbox.TextBoxFormat.FlipHorizontal = true;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends new textbox to the paragraph
```

```

IWTextBox textbox = paragraph.AppendTextBox(150, 75);
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
textboxParagraph.AppendText("Text inside text box");
//Sets fill color and line width for textbox
textbox.TextBoxFormat.FillColor = Syncfusion.Drawing.Color.LightGreen;
textbox.TextBoxFormat.LineWidth = 2;
//Applies textbox text direction
textbox.TextBoxFormat.TextDirection =
Syncfusion.DocIO.DLS.TextDirection.VerticalTopToBottom;
//Sets text wrapping style
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical position
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Margin;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Sets top and bottom margin values
textbox.TextBoxFormat.InternalMargin.Bottom = 5f;
textbox.TextBoxFormat.InternalMargin.Top = 5f;
//Sets 90 degree rotation
textbox.TextBoxFormat.Rotation = 90;
//Sets horizontal flip
textbox.TextBoxFormat.FlipHorizontal = true;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

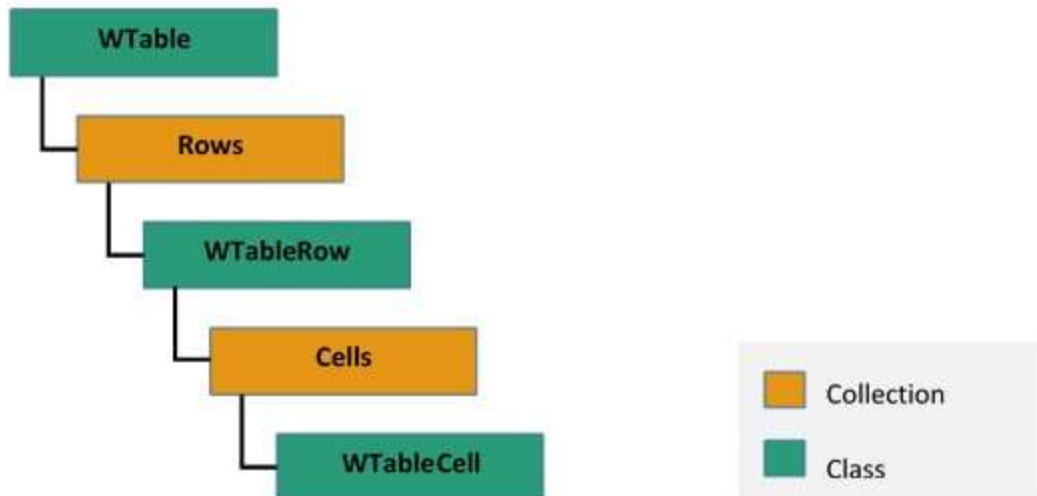
Working with Tables

A table in Word document is used to arrange document content in rows and columns. **WTable** instance represents a table in Word document. A table must contain at least one row.

1. A row is a collection of cells and it is represented by an instance of **WTableRow**. Each row must contain at least one cell.
2. A cell can contain one or more paragraphs and tables. An instance of **WTableCell** represents a table cell. Each table cell must contain at least one paragraph.

Note: Adding more than 63 columns not supported in Word document using Microsoft Word application. It shows alert when you attempt to insert table with more than 64 columns, which is a one of the behaviors of Microsoft Word and Essential DocIO does the same.

The following image illustrates how a table in Word document is organized in EssentialDocIO's DOM:



The following code example illustrates how to create a simple table with predefined number of rows and cells.

C#

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
//Adds a section into Word document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWTextRange textRange = section.AddParagraph().AppendText("Price Details");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 12;
textRange.CharacterFormat.Bold = true;
section.AddParagraph();
//Adds a new table into Word document
IWTable table = section.AddTable();
//Specifies the total number of rows & columns
table.ResetCells(3, 2);
//Accesses the instance of the cell (first row, first cell) and adds the
content into cell
textRange = table[0, 0].AddParagraph().AppendText("Item");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 12;
textRange.CharacterFormat.Bold = true;
//Accesses the instance of the cell (first row, second cell) and adds the
content into cell
textRange = table[0, 1].AddParagraph().AppendText("Price($)");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 12;
textRange.CharacterFormat.Bold = true;
//Accesses the instance of the cell (second row, first cell) and adds the
content into cell
textRange = table[1, 0].AddParagraph().AppendText("Apple");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Accesses the instance of the cell (second row, second cell) and adds the
content into cell
textRange = table[1, 1].AddParagraph().AppendText("50");
  
```

```

textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Accesses the instance of the cell (third row, first cell) and adds the content into cell
textRange = table[2, 0].AddParagraph().AppendText("Orange");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Accesses the instance of the cell (third row, second cell) and adds the content into cell
textRange = table[2, 1].AddParagraph().AppendText("30");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Saves the document in the given name and format
document.Save("Table.docx", FormatType.Docx);
//Releases the resources occupied by WordDocument instance
document.Close();

```

VB.NET

```

'Creates an instance of WordDocument class
Dim document As New WordDocument()
'Adds a section into Word document
Dim section As IWSection = document.AddSection()
'Adds a new paragraph into Word document and appends text into paragraph
Dim textRange As IWTextRange = section.AddParagraph().AppendText("Price Details")
textRange.CharacterFormat.FontName = "Arial"
textRange.CharacterFormat.FontSize = 12
textRange.CharacterFormat.Bold = True
section.AddParagraph()
'Adds a new table into Word document
Dim table As IWTable = section.AddTable()
'Specifies the total number of rows and columns
table.ResetCells(3, 2)
'Accesses the instance of the cell (first row, first cell) and adds the content into cell
textRange = table(0, 0).AddParagraph().AppendText("Item")
textRange.CharacterFormat.FontName = "Arial"
textRange.CharacterFormat.FontSize = 12
textRange.CharacterFormat.Bold = True
'Accesses the instance of the cell (first row, second cell) and adds the content into cell
textRange = table(0, 1).AddParagraph().AppendText("Price ($) ")
textRange.CharacterFormat.FontName = "Arial"
textRange.CharacterFormat.FontSize = 12
textRange.CharacterFormat.Bold = True
'Accesses the instance of the cell (second row, first cell) and adds the content into cell
textRange = table(1, 0).AddParagraph().AppendText("Apple")
textRange.CharacterFormat.FontName = "Arial"
textRange.CharacterFormat.FontSize = 10
'Accesses the instance of the cell (second row, second cell) and adds the content into cell
textRange = table(1, 1).AddParagraph().AppendText("50")
textRange.CharacterFormat.FontName = "Arial"
textRange.CharacterFormat.FontSize = 10

```

```

'Accesses the instance of the cell (third row, first cell) and adds the content into cell
textRange = table(2, 0).AddParagraph().AppendText("Orange")
textRange.CharacterFormat.FontName = "Arial"
textRange.CharacterFormat.FontSize = 10
'Accesses the instance of the cell (third row, second cell) and adds the content into cell
textRange = table(2, 1).AddParagraph().AppendText("30")
textRange.CharacterFormat.FontName = "Arial"
textRange.CharacterFormat.FontSize = 10
'Saves the document in the given name and format
document.Save("Table.docx", FormatType.Docx)
'Releases the resources occupied by WordDocument instance
document.Close()

```

UWP

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
//Adds a section into Word document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWTextRange textRange = section.AddParagraph().AppendText("Price Details");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 12;
textRange.CharacterFormat.Bold = true;
section.AddParagraph();
//Adds a new table into Word document
IWTable table = section.AddTable();
//Specifies the total number of rows & columns
table.ResetCells(3, 2);
//Accesses the instance of the cell (first row, first cell) and adds the content into cell
textRange = table[0, 0].AddParagraph().AppendText("Item");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 12;
textRange.CharacterFormat.Bold = true;
//Accesses the instance of the cell (first row, second cell) and adds the content into cell
textRange = table[0, 1].AddParagraph().AppendText("Price ($)");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 12;
textRange.CharacterFormat.Bold = true;
//Accesses the instance of the cell (second row, first cell) and adds the content into cell
textRange = table[1, 0].AddParagraph().AppendText("Apple");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Accesses the instance of the cell (second row, second cell) and adds the content into cell
textRange = table[1, 1].AddParagraph().AppendText("50");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Accesses the instance of the cell (third row, first cell) and adds the content into cell
textRange = table[2, 0].AddParagraph().AppendText("Orange");

```

```

textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Accesses the instance of the cell (third row, second cell) and adds the content into cell
textRange = table[2, 1].AddParagraph().AppendText("30");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Table.docx");
//Releases the resources occupied by WordDocument instance
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
//Adds a section into Word document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWTextRange textRange = section.AddParagraph().AppendText("Price Details");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 12;
textRange.CharacterFormat.Bold = true;
section.AddParagraph();
//Adds a new table into Word document
IWTable table = section.AddTable();
//Specifies the total number of rows & columns
table.ResetCells(3, 2);
//Accesses the instance of the cell (first row, first cell) and adds the content into cell
textRange = table[0, 0].AddParagraph().AppendText("Item");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 12;
textRange.CharacterFormat.Bold = true;
//Accesses the instance of the cell (first row, second cell) and adds the content into cell
textRange = table[0, 1].AddParagraph().AppendText("Price($)");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 12;
textRange.CharacterFormat.Bold = true;
//Accesses the instance of the cell (second row, first cell) and adds the content into cell
textRange = table[1, 0].AddParagraph().AppendText("Apple");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Accesses the instance of the cell (second row, second cell) and adds the content into cell
textRange = table[1, 1].AddParagraph().AppendText("50");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;

```

```
//Accesses the instance of the cell (third row, first cell) and adds the
content into cell
textRange = table[2, 0].AddParagraph().AppendText("Orange");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Accesses the instance of the cell (third row, second cell) and adds the
content into cell
textRange = table[2, 1].AddParagraph().AppendText("30");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Table.docx");
```

XAMARIN

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
//Adds a section into Word document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWTextRange textRange = section.AddParagraph().AppendText("Price Details");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 12;
textRange.CharacterFormat.Bold = true;
section.AddParagraph();
//Adds a new table into Word document
IWTable table = section.AddTable();
//Specifies the total number of rows & columns
table.ResetCells(3, 2);
//Accesses the instance of the cell (first row, first cell) and adds the
content into cell
textRange = table[0, 0].AddParagraph().AppendText("Item");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 12;
textRange.CharacterFormat.Bold = true;
//Accesses the instance of the cell (first row, second cell) and adds the
content into cell
textRange = table[0, 1].AddParagraph().AppendText("Price($)");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 12;
textRange.CharacterFormat.Bold = true;
//Accesses the instance of the cell (second row, first cell) and adds the
content into cell
textRange = table[1, 0].AddParagraph().AppendText("Apple");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Accesses the instance of the cell (second row, second cell) and adds the
content into cell
textRange = table[1, 1].AddParagraph().AppendText("50");
textRange.CharacterFormat.FontName = "Arial";
```

```

textRange.CharacterFormat.FontSize = 10;
//Accesses the instance of the cell (third row, first cell) and adds the
content into cell
textRange = table[2, 0].AddParagraph().AppendText("Orange");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Accesses the instance of the cell (third row, second cell) and adds the
content into cell
textRange = table[2, 1].AddParagraph().AppendText("30");
textRange.CharacterFormat.FontName = "Arial";
textRange.CharacterFormat.FontSize = 10;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Table.docx",
"application/msword", stream);
//Releases the resources occupied by WordDocument instance
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example illustrates how to create a simple table by dynamically adding rows.

C#

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Price Details");
section.AddParagraph();
//Adds a new table into Word document
IWTable table = section.AddTable();
//Adds the first row into table
WTableRow row = table.AddRow();
//Adds the first cell into first row
WTableCell cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Item");
//Adds the second cell into first row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Price($)");
//Adds the second row into table
row = table.AddRow(true, false);
//Adds the first cell into second row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Apple");
//Adds the second cell into second row
cell = row.AddCell();

```



```

//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("50");
//Adds the third row into table
row = table.AddRow(true, false);
//Adds the first cell into third row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Orange");
//Adds the second cell into third row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("30");
//Adds the fourth row into table
row = table.AddRow(true, false);
//Adds the first cell into fourth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Banana");
//Adds the second cell into fourth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("20");
//Adds the fifth row to table
row = table.AddRow(true, false);
//Adds the first cell into fifth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Grapes");
//Adds the second cell into fifth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("70");
//Saves and closes the document instance
document.Save("Table.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates an instance of WordDocument class
Dim document As New WordDocument()
Dim section As IWSection = document.AddSection()
section.AddParagraph().AppendText("Price Details")
section.AddParagraph()
'Adds a new table into Word document
Dim table As IWTable = section.AddTable()
'Adds the first row into table
Dim row As WTableRow = table.AddRow()
'Adds the first cell into first row
Dim cell As WTableCell = row.AddCell()

```

```
'Specifies the cell width
cell.Width = 200
cell.AddParagraph().AppendText("Item")
'Adds the second cell into first row
cell = row.AddCell()
'Specifies the cell width
cell.Width = 200
cell.AddParagraph().AppendText("Price($) ")
'Adds the second row into table
row = table.AddRow(True, False)
'Adds the first cell into second row
cell = row.AddCell()
'Specifies the cell width
cell.Width = 200
cell.AddParagraph().AppendText("Apple")
'Adds the second cell into second row
cell = row.AddCell()
'Specifies the cell width
cell.Width = 200
cell.AddParagraph().AppendText("50")
'Adds the third row into table
row = table.AddRow(True, False)
'Adds the first cell into third row
cell = row.AddCell()
'Specifies the cell width
cell.Width = 200
cell.AddParagraph().AppendText("Orange")
'Adds the second cell into third row
cell = row.AddCell()
'Specifies the cell width
cell.Width = 200
cell.AddParagraph().AppendText("30")
'Adds the fourth row into table
row = table.AddRow(True, False)
'Adds the first cell into fourth row
cell = row.AddCell()
'Specifies the cell width
cell.Width = 200
cell.AddParagraph().AppendText("Banana")
'Adds the second cell into fourth row
cell = row.AddCell()
'Specifies the cell width
cell.Width = 200
cell.AddParagraph().AppendText("20")
'Adds the fifth row to table
row = table.AddRow(True, False)
'Adds the first cell into fifth row
cell = row.AddCell()
'Specifies the cell width
cell.Width = 200
cell.AddParagraph().AppendText("Grapes")
'Adds the second cell into fifth row
cell = row.AddCell()
'Specifies the cell width
cell.Width = 200
cell.AddParagraph().AppendText("70")
'Saves and closes the document instance
```

```
document.Save("Table.docx", FormatType.Docx);
document.Close();
```

UWP

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Price Details");
section.AddParagraph();
//Adds a new table into Word document
IWTable table = section.AddTable();
//Adds the first row into table
WTableRow row = table.AddRow();
//Adds the first cell into first row
WTableCell cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Item");
//Adds the second cell into first row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Price($)");
//Adds the second row into table
row = table.AddRow(true, false);
//Adds the first cell into second row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Apple");
//Adds the second cell into second row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("50");
//Adds the third row into table
row = table.AddRow(true, false);
//Adds the first cell into third row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Orange");
//Adds the second cell into third row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("30");
//Adds the fourth row into table
row = table.AddRow(true, false);
//Adds the first cell into fourth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Banana");
//Adds the second cell into fourth row
```

```

cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("20");
//Adds the fifth row to table
row = table.AddRow(true, false);
//Adds the first cell into fifth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Grapes");
//Adds the second cell into fifth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("70");
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Table.docx");
//Closes the document instance
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Price Details");
section.AddParagraph();
//Adds a new table into Word document
IWTable table = section.AddTable();
//Adds the first row into table
WTableRow row = table.AddRow();
//Adds the first cell into first row
WTableCell cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Item");
//Adds the second cell into first row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Price($)");
//Adds the second row into table
row = table.AddRow(true, false);
//Adds the first cell into second row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Apple");
//Adds the second cell into second row

```

```

cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("50");
//Adds the third row into table
row = table.AddRow(true, false);
//Adds the first cell into third row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Orange");
//Adds the second cell into third row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("30");
//Adds the fourth row into table
row = table.AddRow(true, false);
//Adds the first cell into fourth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Banana");
//Adds the second cell into fourth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("20");
//Adds the fifth row to table
row = table.AddRow(true, false);
//Adds the first cell into fifth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Grapes");
//Adds the second cell into fifth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("70");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Table.docx");

```

XAMARIN

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Price Details");
section.AddParagraph();

```

```
//Adds a new table into Word document
IWTable table = section.AddTable();
//Adds the first row into table
WTableRow row = table.AddRow();
//Adds the first cell into first row
WTableCell cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Item");
//Adds the second cell into first row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Price($)");
//Adds the second row into table
row = table.AddRow(true, false);
//Adds the first cell into second row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Apple");
//Adds the second cell into second row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("50");
//Adds the third row into table
row = table.AddRow(true, false);
//Adds the first cell into third row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Orange");
//Adds the second cell into third row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("30");
//Adds the fourth row into table
row = table.AddRow(true, false);
//Adds the first cell into fourth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Banana");
//Adds the second cell into fourth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("20");
//Adds the fifth row to table
row = table.AddRow(true, false);
//Adds the first cell into fifth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("Grapes");
```

```
//Adds the second cell into fifth row
cell = row.AddCell();
//Specifies the cell width
cell.Width = 200;
cell.AddParagraph().AppendText("70");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Table.docx",
"application/msword", stream);
//Closes the document instance
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Nested Table

You can create a nested table by adding a new table into a cell. The following code example illustrates how to add a table into a cell.

C#

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Price Details");
IWTable table = section.AddTable();
table.ResetCells(3, 2);
table[0, 0].AddParagraph().AppendText("Item");
table[0, 1].AddParagraph().AppendText("Price($)");
table[1, 0].AddParagraph().AppendText("Items with same price");
//Adds a nested table into the cell (second row, first cell).
IWTable nestTable = table[1, 0].AddTable();
//Creates the specified number of rows and columns to nested table
nestTable.ResetCells(3, 1);
//Accesses the instance of the nested table cell (first row, first cell)
WTableCell nestedCell = nestTable.Rows[0].Cells[0];
//Specifies the width of the nested cell
nestedCell.Width = 200;
//Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Apple");
//Accesses the instance of the nested table cell (second row, first cell)
nestedCell = nestTable.Rows[1].Cells[0];
//Specifies the width of the nested cell
nestedCell.Width = 200;
//Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Orange");
//Accesses the instance of the nested table cell (third row, first cell)
nestedCell = nestTable.Rows[2].Cells[0];
//Specifies the width of the nested cell
nestedCell.Width = 200;
//Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Mango");
//Accesses the instance of the cell (second row, second cell)
```

```
nestedCell = table.Rows[1].Cells[1];
table[1, 1].AddParagraph().AppendText("85");
table[2, 0].AddParagraph().AppendText("Pomegranate");
table[2, 1].AddParagraph().AppendText("70");
//Saves and closes the document instance
document.Save("NestedTable.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates an instance of WordDocument class
Dim document As New WordDocument()
Dim section As IWSection = document.AddSection()
section.AddParagraph().AppendText("Price Details")
Dim table As ITable = section.AddTable()
table.ResetCells(3, 2)
table(0, 0).AddParagraph().AppendText("Item")
table(0, 1).AddParagraph().AppendText("Price($)")
table(1, 0).AddParagraph().AppendText("Items with same price")
'Adds a nested table into the cell (second row, first cell).
Dim nestTable As ITable = table(1, 0).AddTable()
'Creates the specified number of rows and columns to nested table
nestTable.ResetCells(3, 1)
'Accesses the instance of the nested table cell (first row, first cell)
Dim nestedCell As WTableCell = nestTable.Rows(0).Cells(0)
'Specifies the width of the nested cell
nestedCell.Width = 200
'Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Apple")
'Accesses the instance of the nested table cell (second row, first cell)
nestedCell = nestTable.Rows(1).Cells(0)
'Specifies the width of the nested cell
nestedCell.Width = 200
'Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Orange")
'Accesses the instance of the nested table cell (third row, first cell)
nestedCell = nestTable.Rows(2).Cells(0)
'Specifies the width of the nested cell
nestedCell.Width = 200
'Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Mango")
'Accesses the instance of the cell (second row, second cell)
nestedCell = table.Rows(1).Cells(1)
table(1, 1).AddParagraph().AppendText("85")
table(2, 0).AddParagraph().AppendText("Pomegranate")
table(2, 1).AddParagraph().AppendText("70")
'Saves and closes the document instance
document.Save("NestedTable.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Price Details");
```



```

IWTable table = section.AddTable();
table.ResetCells(3, 2);
table[0, 0].AddParagraph().AppendText("Item");
table[0, 1].AddParagraph().AppendText("Price($)");
table[1, 0].AddParagraph().AppendText("Items with same price");
//Adds a nested table into the cell (second row, first cell).
IWTable nestTable = table[1, 0].AddTable();
//Creates the specified number of rows and columns to nested table
nestTable.ResetCells(3, 1);
//Accesses the instance of the nested table cell (first row, first cell)
WTableCell nestedCell = nestTable.Rows[0].Cells[0];
//Specifies the width of the nested cell
nestedCell.Width = 200;
//Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Apple");
//Accesses the instance of the nested table cell (second row, first cell)
nestedCell = nestTable.Rows[1].Cells[0];
//Specifies the width of the nested cell
nestedCell.Width = 200;
//Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Orange");
//Accesses the instance of the nested table cell (third row, first cell)
nestedCell = nestTable.Rows[2].Cells[0];
//Specifies the width of the nested cell
nestedCell.Width = 200;
//Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Mango");
//Accesses the instance of the cell (second row, second cell)
nestedCell = table.Rows[1].Cells[1];
table[1, 1].AddParagraph().AppendText("85");
table[2, 0].AddParagraph().AppendText("Pomegranate");
table[2, 1].AddParagraph().AppendText("70");
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "NestedTable.docx");
//Closes the document instance
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Price Details");
IWTable table = section.AddTable();
table.ResetCells(3, 2);
table[0, 0].AddParagraph().AppendText("Item");
table[0, 1].AddParagraph().AppendText("Price($)");
table[1, 0].AddParagraph().AppendText("Items with same price");
//Adds a nested table into the cell (second row, first cell).
IWTable nestTable = table[1, 0].AddTable();

```

```
//Creates the specified number of rows and columns to nested table
nestTable.ResetCells(3, 1);
//Accesses the instance of the nested table cell (first row, first cell)
WTableCell nestedCell = nestTable.Rows[0].Cells[0];
//Specifies the width of the nested cell
nestedCell.Width = 200;
//Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Apple");
//Accesses the instance of the nested table cell (second row, first cell)
nestedCell = nestTable.Rows[1].Cells[0];
//Specifies the width of the nested cell
nestedCell.Width = 200;
//Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Orange");
//Accesses the instance of the nested table cell (third row, first cell)
nestedCell = nestTable.Rows[2].Cells[0];
//Specifies the width of the nested cell
nestedCell.Width = 200;
//Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Mango");
//Accesses the instance of the cell (second row, second cell)
nestedCell = table.Rows[1].Cells[1];
table[1, 1].AddParagraph().AppendText("85");
table[2, 0].AddParagraph().AppendText("Pomegranate");
table[2, 1].AddParagraph().AppendText("70");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "NestedTable.docx");
```

XAMARIN

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Price Details");
IWTable table = section.AddTable();
table.ResetCells(3, 2);
table[0, 0].AddParagraph().AppendText("Item");
table[0, 1].AddParagraph().AppendText("Price($)");
table[1, 0].AddParagraph().AppendText("Items with same price");
//Adds a nested table into the cell (second row, first cell).
IWTable nestTable = table[1, 0].AddTable();
//Creates the specified number of rows and columns to nested table
nestTable.ResetCells(3, 1);
//Accesses the instance of the nested table cell (first row, first cell)
WTableCell nestedCell = nestTable.Rows[0].Cells[0];
//Specifies the width of the nested cell
nestedCell.Width = 200;
//Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Apple");
//Accesses the instance of the nested table cell (second row, first cell)
```

```

nestedCell = nestTable.Rows[1].Cells[0];
//Specifies the width of the nested cell
nestedCell.Width = 200;
//Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Orange");
//Accesses the instance of the nested table cell (third row, first cell)
nestedCell = nestTable.Rows[2].Cells[0];
//Specifies the width of the nested cell
nestedCell.Width = 200;
//Adds the content into nested cell
nestedCell.AddParagraph().AppendText("Mango");
//Accesses the instance of the cell (second row, second cell)
nestedCell = table.Rows[1].Cells[1];
table[1, 1].AddParagraph().AppendText("85");
table[2, 0].AddParagraph().AppendText("Pomegranate");
table[2, 1].AddParagraph().AppendText("70");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("NestedTable.docx",
"application/msword", stream);
//Closes the document instance
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Align text within a table

You can iterate the cells within a table and align text for each cell. Find more information about iterating the cells from [here](#)

The following code example illustrates how to align text within a table.

C#

```

private void AlignCellContent(WTableCell tableCell, VerticalAlignment
verticalAlignment, HorizontalAlignment horizontalAlignment)
{
    //Sets vertical alignment to the cell.
    tableCell.CellFormat.VerticalAlignment = verticalAlignment;
    //Iterates body items in table cell and set horizontal alignment.
    AlignCellContentForTextBody(tableCell, horizontalAlignment);
}
private void AlignCellContentForTextBody(WTextBody textBody,
HorizontalAlignment horizontalAlignment)
{
    for (int i = 0; i < textBody.ChildEntities.Count; i++)
    {
        //IEntity is the basic unit in DocIO DOM.
        //Accesses the body items as IEntity
        IEntity bodyItemEntity = textBody.ChildEntities[i];
        //A Text body has 3 types of elements - Paragraph, Table and Block Content
        Control
        //Decides the element type by using EntityType
    }
}

```

```

switch (bodyItemEntity.EntityType)
{
    case EntityType.Paragraph:
        WParagraph paragraph = bodyItemEntity as WParagraph;
        //Sets horizontal alignment for paragraph.
        paragraph.ParagraphFormat.HorizontalAlignment = horizontalAlignment;
        break;
    case EntityType.Table:
        //Table is a collection of rows and cells
        //Iterates through table's DOM and set horizontal alignment.
        AlignCellContentForTable(bodyItemEntity as WTable, horizontalAlignment);
        break;
    case EntityType.BlockContentControl:
        BlockContentControl blockContentControl = bodyItemEntity as
        BlockContentControl;
        //Iterates to the body items of Block Content Control and set horizontal
        alignment.
        AlignCellContentForTextBody(blockContentControl.TextBody,
        horizontalAlignment);
        break;
}
}
}

private void AlignCellContentForTable(WTable table,
Syncfusion.DocIO.DLS.HorizontalAlignment horizontalAlignment)
{
    //Iterates the row collection in a table
    foreach (WTableRow row in table.Rows)
    {
        //Iterates the cell collection in a table row
        foreach (WTableCell cell in row.Cells)
        {
            //Iterate items in cell and set horizontal alignment
            AlignCellContentForTextBody(cell, horizontalAlignment);
        }
    }
}

```

Apply formatting to Table, Row and Cell

The following code example illustrates how to load an existing document and apply table formatting options such as Borders, LeftIndent, Paddings, IsAutoResize, etc.

Note: 1. `BorderStyle.None` is the default value of `BorderType` property in `Borders` class which will not show borders for the table or cell. It is equivalent to border style not defined and borders can be inherited from style or parent formats.

2. To hide border for a table or cell in the Word Document, you can set `BorderType` property with `BorderStyle.Cleared`. It means border style defined as no border (Don't show border) and shouldn't inherit from style or parent formats.

3. To show/display border for a table or cell in the Word Document, you can set `BorderType` property with `BorderStyle` values (except `BorderStyle.None` and `BorderStyle.Cleared`).

4. As in the Microsoft Word, DocIO supports `RowFormat.Borders` in DOC format alone.

C#

```

//Creates an instance of WordDocument class (Empty Word Document)
WordDocument document = new WordDocument();
//Opens an existing Word document into DocIO instance
document.Open("Table.docx", FormatType.Docx);
//Accesses the instance of the first section in the Word document
WSection section = document.Sections[0];
//Accesses the instance of the first table in the section
WTable table = section.Tables[0] as WTable;
//Specifies the title for the table
table.Title = "PriceDetails";
//Specifies the description of the table
table.Description = "This table shows the price details of various fruits";
//Specifies the left indent of the table
table.IndentFromLeft = 50;
//Specifies the background color of the table
table.TableFormat.BackColor = Color.FromArgb(192, 192, 192);
//Specifies the horizontal alignment of the table
table.TableFormat.HorizontalAlignment = RowAlignment.Left;
//Specifies the left, right, top and bottom padding of all the cells in the table
table.TableFormat.Paddings.All = 10;
//Specifies the auto resize of table to automatically resize all cell width based on its content
table.TableFormat.IsAutoResized = true;
//Specifies the table top, bottom, left and right border line width
table.TableFormat.Borders.LineWidth = 2f;
//Specifies the table horizontal border line width
table.TableFormat.Borders.Horizontal.LineWidth = 2f;
//Specifies the table vertical border line width
table.TableFormat.Borders.Vertical.LineWidth = 2f;
//Specifies the tables top, bottom, left and right border color
table.TableFormat.Borders.Color = Color.Red;
//Specifies the table Horizontal border color
table.TableFormat.Borders.Horizontal.Color = Color.Red;
//Specifies the table vertical border color
table.TableFormat.Borders.Vertical.Color = Color.Red;
//Specifies the table borders border type
table.TableFormat.Borders.BorderType = BorderStyle.Double;
//Accesses the instance of the first row in the table
WTableRow row = table.Rows[0];
//Specifies the row height
row.Height = 20;
//Specifies the row height type
row.HeightType = TableRowHeightType.AtLeast;
//Saves and closes the document instance
document.Save("TableFormatting.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates an instance of WordDocument class (Empty Word Document)
Dim document As New WordDocument()
'Opens an existing Word document into DocIO instance
document.Open("Table.docx", FormatType.Docx)

```

```

'Accesses the instance of the first section in the Word document
Dim section As WSection = document.Sections(0)
'Accesses the instance of the first table in the section
Dim table As WTable = TryCast(section.Tables(0), WTable)
'Specifies the title for the table
table.Title = "PriceDetails"
'Specifies the description of the table
table.Description = "This table shows the price details of various fruits"
'Specifies the left indent of the table
table.IndentFromLeft = 50
'Specifies the background color of the table
table.TableFormat.BackColor = Color.FromArgb(192, 192, 192)
'Specifies the horizontal alignment of the table
table.TableFormat.HorizontalAlignment = RowAlignment.Left
'Specifies the left, right, top and bottom padding of all the cells in the table
table.TableFormat.Paddings.All = 10
'Specifies the auto resize of table to automatically resize all cell width based on its content
table.TableFormat.IsAutoResized = True
'Specifies the table top, bottom, left and right border line width
table.TableFormat.Borders.LineWidth = 2.0F
'Specifies the table horizontal border line width
table.TableFormat.Borders.Horizontal.LineWidth = 2.0F
'Specifies the table vertical border line width
table.TableFormat.Borders.Vertical.LineWidth = 2.0F
'Specifies the tables top, bottom, left and right border color
table.TableFormat.Borders.Color = Color.Red
'Specifies the table Horizontal border color
table.TableFormat.Borders.Horizontal.Color = Color.Red
'Specifies the table vertical border color
table.TableFormat.Borders.Vertical.Color = Color.Red
'Specifies the table borders border type
table.TableFormat.Borders.BorderType = BorderStyle.[Double]
'Access the instance of the first row in the table
Dim row As WTableRow = table.Rows(0)
'Specifies the row height
row.Height = 20
'Specifies the row height type
row.HeightType = TableRowHeightType.AtLeast
'Saves and closes the document instance
document.Save("TableFormatting.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates an instance of WordDocument class (Empty Word Document)
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new WordDocument();
//Opens an existing Word document into DocIO instance
document.Open(assembly.GetManifestResourceStream("Sample.Assets.Table.docx"),
, FormatType.Docx);
//Accesses the instance of the first section in the Word document
WSection section = document.Sections[0];
//Accesses the instance of the first table in the section
WTable table = section.Tables[0] as WTable;

```

```

//Specifies the title for the table
table.Title = "PriceDetails";
//Specifies the description of the table
table.Description = "This table shows the price details of various fruits";
//Specifies the left indent of the table
table.IndentFromLeft = 50;
//Specifies the background color of the table
table.TableFormat.BackColor = Color.FromArgb(192, 192, 192);
//Specifies the horizontal alignment of the table
table.TableFormat.HorizontalAlignment = RowAlignment.Left;
//Specifies the left, right, top and bottom padding of all the cells in the table
table.TableFormat.Paddings.All = 10;
//Specifies the auto resize of table to automatically resize all cell width based on its content
table.TableFormat.IsAutoResized = true;
//Specifies the table top, bottom, left and right border line width
table.TableFormat.Borders.LineWidth = 2f;
//Specifies the table horizontal border line width
table.TableFormat.Borders.Horizontal.LineWidth = 2f;
//Specifies the table vertical border line width
table.TableFormat.Borders.Vertical.LineWidth = 2f;
//Specifies the tables top, bottom, left and right border color
table.TableFormat.Borders.Color = Color.Red;
//Specifies the table Horizontal border color
table.TableFormat.Borders.Horizontal.Color = Color.Red;
//Specifies the table vertical border color
table.TableFormat.Borders.Vertical.Color = Color.Red;
//Specifies the table borders border type
table.TableFormat.Borders.BorderType = BorderStyle.Double;
//Accesses the instance of the first row in the table
WTableRow row = table.Rows[0];
//Specifies the row height
row.Height = 20;
//Specifies the row height type
row.HeightType = TableRowHeightType.AtLeast;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "TableFormatting.docx");
//Closes the document instance
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class (Empty Word Document)
FileStream fileStreamPath = new FileStream("Table.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument();
//Opens an existing Word document into DocIO instance
document.Open(fileStreamPath, FormatType.Docx);
//Accesses the instance of the first section in the Word document

```

```

WSection section = document.Sections[0];
//Accesses the instance of the first table in the section
WTable table = section.Tables[0] as WTable;
//Specifies the title for the table
table.Title = "PriceDetails";
//Specifies the description of the table
table.Description = "This table shows the price details of various fruits";
//Specifies the left indent of the table
table.IndentFromLeft = 50;
//Specifies the background color of the table
table.TableFormat.BackColor = Color.FromArgb(192, 192, 192);
//Specifies the horizontal alignment of the table
table.TableFormat.HorizontalAlignment = RowAlignment.Left;
//Specifies the left, right, top and bottom padding of all the cells in the
table
table.TableFormat.Paddings.All = 10;
//Specifies the auto resize of table to automatically resize all cell width
based on its content
table.TableFormat.IsAutoResized = true;
//Specifies the table top, bottom, left and right border line width
table.TableFormat.Borders.LineWidth = 2f;
//Specifies the table horizontal border line width
table.TableFormat.Borders.Horizontal.LineWidth = 2f;
//Specifies the table vertical border line width
table.TableFormat.Borders.Vertical.LineWidth = 2f;
//Specifies the tables top, bottom, left and right border color
table.TableFormat.Borders.Color = Color.Red;
//Specifies the table Horizontal border color
table.TableFormat.Borders.Horizontal.Color = Color.Red;
//Specifies the table vertical border color
table.TableFormat.Borders.Vertical.Color = Color.Red;
//Specifies the table borders border type
table.TableFormat.Borders.BorderType = BorderStyle.Double;
//Accesses the instance of the first row in the table
WTableRow row = table.Rows[0];
//Specifies the row height
row.Height = 20;
//Specifies the row height type
row.HeightType = TableRowHeightType.AtLeast;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "TableFormatting.docx");

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Creates an instance of WordDocument class (Empty Word Document)
WordDocument document = new WordDocument();
//Opens an existing Word document into DocIO instance
document.Open(assembly.GetManifestResourceStream("GettingStarted.Data.Table.
docx"), FormatType.Docx);

```



```

//Accesses the instance of the first section in the Word document
WSection section = document.Sections[0];
//Accesses the instance of the first table in the section
WTable table = section.Tables[0] as WTable;
//Specifies the title for the table
table.Title = "PriceDetails";
//Specifies the description of the table
table.Description = "This table shows the price details of various fruits";
//Specifies the left indent of the table
table.IndentFromLeft = 50;
//Specifies the background color of the table
table.TableFormat.BackColor = Syncfusion.Drawing.Color.FromArgb(192, 192,
192);
//Specifies the horizontal alignment of the table
table.TableFormat.HorizontalAlignment = RowAlignment.Left;
//Specifies the left, right, top and bottom padding of all the cells in the
table
table.TableFormat.Paddings.All = 10;
//Specifies the auto resize of table to automatically resize all cell width
based on its content
table.TableFormat.IsAutoResized = true;
//Specifies the table top, bottom, left and right border line width
table.TableFormat.Borders.LineWidth = 2f;
//Specifies the table horizontal border line width
table.TableFormat.Borders.Horizontal.LineWidth = 2f;
//Specifies the table vertical border line width
table.TableFormat.Borders.Vertical.LineWidth = 2f;
//Specifies the tables top, bottom, left and right border color
table.TableFormat.Borders.Color = Syncfusion.Drawing.Color.Red;
//Specifies the table Horizontal border color
table.TableFormat.Borders.Horizontal.Color = Syncfusion.Drawing.Color.Red;
//Specifies the table vertical border color
table.TableFormat.Borders.Vertical.Color = Syncfusion.Drawing.Color.Red;
//Specifies the table borders border type
table.TableFormat.Borders.BorderType = BorderStyle.Double;
//Accesses the instance of the first row in the table
WTableRow row = table.Rows[0];
//Specifies the row height
row.Height = 20;
//Specifies the row height type
row.HeightType = TableRowHeightType.AtLeast;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TableFormatting.do
cx", "application/msword", stream);
//Closes the document instance
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example illustrates how to load an existing document and apply cell formatting options such as VerticalAlignment, TextDirection, Paddings, Borders, etc.

C#

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
document.Open("Table.docx", FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Accesses the instance of the first row in the table
WTableRow row = table.Rows[0];
//Specifies the row height
row.Height = 20;
//Specifies the row height type
row.HeightType = TableRowHeightType.AtLeast;
//Accesses the instance of the first cell in the row
WTableCell cell = row.Cells[0];
//Specifies the cell back ground color
cell.CellFormat.BackColor = Color.FromArgb(192, 192, 192);
//Specifies the same padding as table option as false to preserve current cell padding
cell.CellFormat.SamePaddingsAsTable = false;
//Specifies the left, right, top and bottom padding of the cell
cell.CellFormat.Paddings.Left = 5;
cell.CellFormat.Paddings.Right = 5;
cell.CellFormat.Paddings.Top = 5;
cell.CellFormat.Paddings.Bottom = 5;
//Specifies the vertical alignment of content of text
cell.CellFormat.VerticalAlignment = VerticalAlignment.Middle;
//Accesses the instance of the second cell in the row
cell = row.Cells[1];
cell.CellFormat.BackColor = Color.FromArgb(192, 192, 192);
cell.CellFormat.SamePaddingsAsTable = false;
//Specifies the left, right, top and bottom padding of the cell
cell.CellFormat.Paddings.All = 5;
cell.CellFormat.VerticalAlignment = VerticalAlignment.Middle;
//Saves and closes the document instance
document.Save("TableCellFormatting.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates an instance of WordDocument class
Dim document As New WordDocument()
document.Open("Table.docx", FormatType.Docx)
Dim section As WSection = document.Sections(0)
Dim table As WTable = TryCast(section.Tables(0), WTable)
'Accesses the instance of the first row in the table
Dim row As WTableRow = table.Rows(0)
'Specifies the row height
row.Height = 20
'Specifies the row height type
row.HeightType = TableRowHeightType.AtLeast
'Accesses the instance of the first cell in the row
Dim cell As WTableCell = row.Cells(0)
'Specifies the cell back ground color
cell.CellFormat.BackColor = Color.FromArgb(192, 192, 192)

```

```

'Specifies the same padding as table option as false to preserve current cell padding
cell.CellFormat.SamePaddingsAsTable = False
'Specifies the left, right, top and bottom padding of the cell
cell.CellFormat.Paddings.Left = 5
cell.CellFormat.Paddings.Right = 5
cell.CellFormat.Paddings.Top = 5
cell.CellFormat.Paddings.Bottom = 5
'Specifies the vertical alignment of content of text
cell.CellFormat.VerticalAlignment = VerticalAlignment.Middle
'Accesses the instance of the second cell in the row
cell = row.Cells(1)
cell.CellFormat.BackColor = Color.FromArgb(192, 192, 192)
cell.CellFormat.SamePaddingsAsTable = False
'Specifies the left, right, top and bottom padding of the cell
cell.CellFormat.Paddings.All = 5
cell.CellFormat.VerticalAlignment = VerticalAlignment.Middle
'Saves and closes the document instance
document.Save("TableCellFormatting.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new WordDocument();
document.Open(assembly.GetManifestResourceStream("Sample.Assets.Table.docx"),
, FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Accesses the instance of the first row in the table
WTableRow row = table.Rows[0];
//Specifies the row height
row.Height = 20;
//Specifies the row height type
row.HeightType = TableRowHeightType.AtLeast;
//Accesses the instance of the first cell in the row
WTableCell cell = row.Cells[0];
//Specifies the cell back ground color
cell.CellFormat.BackColor = Color.FromArgb(192, 192, 192);
//Specifies the same padding as table option as false to preserve current cell padding
cell.CellFormat.SamePaddingsAsTable = false;
//Specifies the left, right, top and bottom padding of the cell
cell.CellFormat.Paddings.Left = 5;
cell.CellFormat.Paddings.Right = 5;
cell.CellFormat.Paddings.Top = 5;
cell.CellFormat.Paddings.Bottom = 5;
//Specifies the vertical alignment of content of text
cell.CellFormat.VerticalAlignment = VerticalAlignment.Middle;
//Accesses the instance of the second cell in the row
cell = row.Cells[1];
cell.CellFormat.BackColor = Color.FromArgb(192, 192, 192);
cell.CellFormat.SamePaddingsAsTable = false;
//Specifies the left, right, top and bottom padding of the cell
cell.CellFormat.Paddings.All = 5;

```

```

cell.CellFormat.VerticalAlignment = VerticalAlignment.Middle;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "TableCellFormatting.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
FileStream fileStreamPath = new FileStream("Table.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
document.Open(fileStreamPath, FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Accesses the instance of the first row in the table
WTableRow row = table.Rows[0];
//Specifies the row height
row.Height = 20;
//Specifies the row height type
row.HeightType = TableRowHeightType.AtLeast;
//Accesses the instance of the first cell in the row
WTableCell cell = row.Cells[0];
//Specifies the cell back ground color
cell.CellFormat.BackColor = Color.FromArgb(192, 192, 192);
//Specifies the same padding as table option as false to preserve current
cell padding
cell.CellFormat.SamePaddingsAsTable = false;
//Specifies the left, right, top and bottom padding of the cell
cell.CellFormat.Paddings.Left = 5;
cell.CellFormat.Paddings.Right = 5;
cell.CellFormat.Paddings.Top = 5;
cell.CellFormat.Paddings.Bottom = 5;
//Specifies the vertical alignment of content of text
cell.CellFormat.VerticalAlignment = VerticalAlignment.Middle;
//Accesses the instance of the second cell in the row
cell = row.Cells[1];
cell.CellFormat.BackColor = Color.FromArgb(192, 192, 192);
cell.CellFormat.SamePaddingsAsTable = false;
//Specifies the left, right, top and bottom padding of the cell
cell.CellFormat.Paddings.All = 5;
cell.CellFormat.VerticalAlignment = VerticalAlignment.Middle;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "TableCellFormatting.docx");

```

XAMARIN

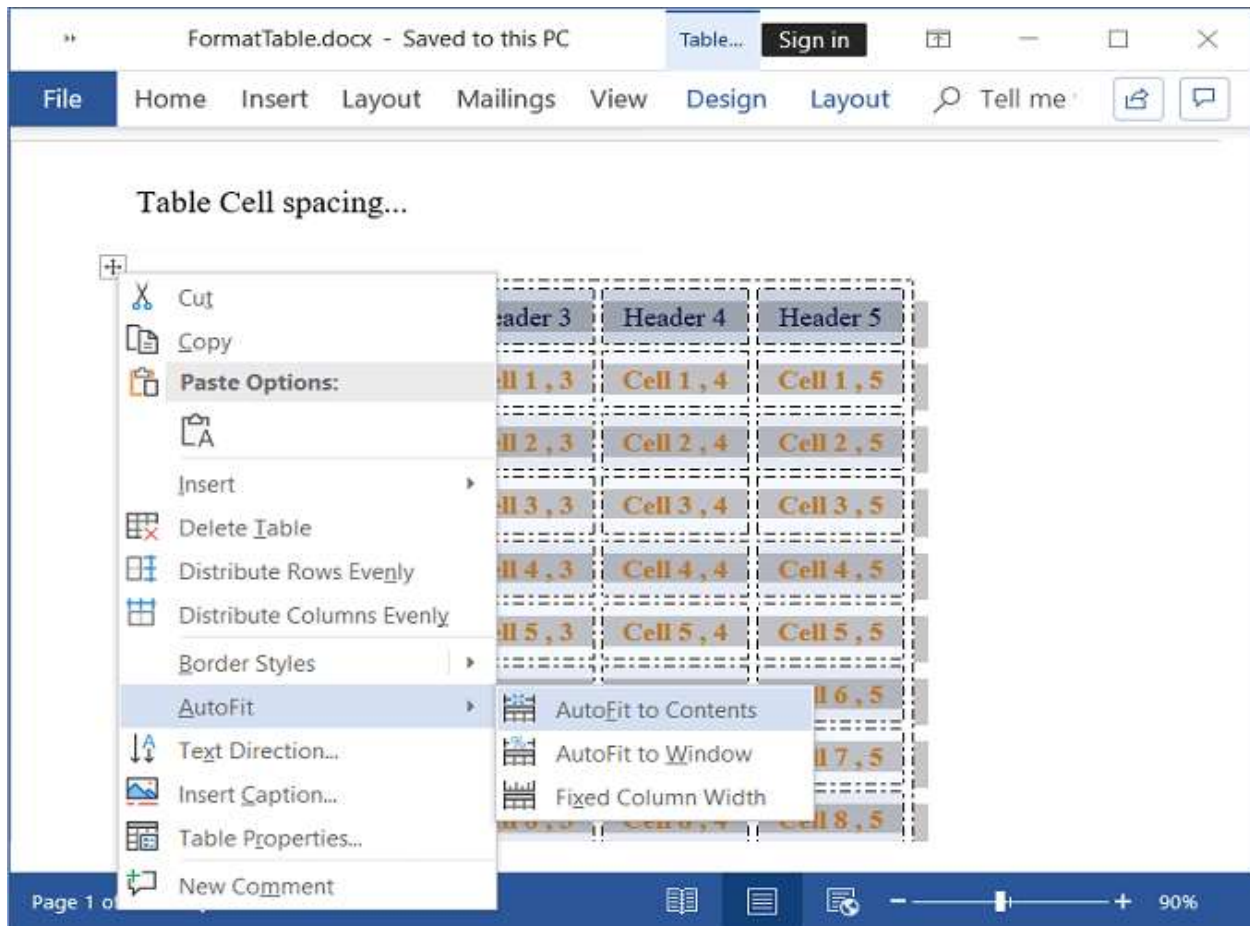
```

//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new WordDocument();
document.Open(assembly.GetManifestResourceStream("GettingStarted.Data.Table.docx"), FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Accesses the instance of the first row in the table
WTableRow row = table.Rows[0];
//Specifies the row height
row.Height = 20;
//Specifies the row height type
row.HeightType = TableRowHeightType.AtLeast;
//Accesses the instance of the first cell in the row
WTableCell cell = row.Cells[0];
//Specifies the cell back ground color
cell.CellFormat.BackColor = Syncfusion.Drawing.Color.FromArgb(192, 192, 192);
//Specifies the same padding as table option as false to preserve current cell padding
cell.CellFormat.SamePaddingsAsTable = false;
//Specifies the left, right, top and bottom padding of the cell
cell.CellFormat.Paddings.Left = 5;
cell.CellFormat.Paddings.Right = 5;
cell.CellFormat.Paddings.Top = 5;
cell.CellFormat.Paddings.Bottom = 5;
//Specifies the vertical alignment of content of text
cell.CellFormat.VerticalAlignment = VerticalAlignment.Middle;
//Accesses the instance of the second cell in the row
cell = row.Cells[1];
cell.CellFormat.BackColor = Syncfusion.Drawing.Color.FromArgb(192, 192, 192);
cell.CellFormat.SamePaddingsAsTable = false;
//Specifies the left, right, top and bottom padding of the cell
cell.CellFormat.Paddings.All = 5;
cell.CellFormat.VerticalAlignment = VerticalAlignment.Middle;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TableCellFormatting.docx", "application/msword", stream);
//Closes the document instance
document.Close();
//Please download the helper files from the below link to save the stream as file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin

```

Resize table

You can automatically resize the table cell to fit its contents based on the given **autofit options** such as **FitToContent**, **FitToWindow**, **FixedColumnWidth**.



The following code example shows how to resize the table in a Word document.

C#

```
//Creates an instance of WordDocument class (Empty Word Document)*'|
markdownify }}
WordDocument document = new WordDocument();
//Opens an existing Word document into DocIO instance
document.Open("Template", FormatType.Docx);
//Accesses the instance of the first section in the Word document
WSection section = document.Sections[0];
//Accesses the instance of the first table in the section
WTable table = section.Tables[0] as WTable;
//Resizes the table to fit the contents respect to the contents
table.AutoFit(AutoFitType.FitToContent);
//Accesses the instance of the second table in the section
table = section.Tables[1] as WTable;
//Resizes the table to fit the contents respect to window/page width
table.AutoFit(AutoFitType.FitToWindow);
//Accesses the instance of the third table in the section
table = section.Tables[2] as WTable;
//Resizes the table to fit the contents respect to fixed column width
table.AutoFit(AutoFitType.FixedColumnWidth);
//Saves and closes the document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```

'Creates an instance of WordDocument class (Empty Word Document)
Dim document As WordDocument = New WordDocument
'Opens an existing Word document into DocIO instance
document.Open("Template", FormatType.Docx)
Dim section As WSection = document.Sections(0)
Dim table As WTable = CType(section.Tables(0), WTable)
'Resizes the table to fit the contents respect to the contents
table.AutoFit(AutoFitType.FitToContent)
'Accesses the instance of the second table in the section
table = CType(section.Tables(1), WTable)
'Resizes the table to fit the contents respect to window/page width
table.AutoFit(AutoFitType.FitToWindow)
'Accesses the instance of the third table in the section
table = CType(section.Tables(2), WTable)
'Resizes the table to fit the contents respect to fixed column width
table.AutoFit(AutoFitType.FixedColumnWidth)
'Saves and closes the document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Creates an instance of WordDocument class (Empty Word Document)
WordDocument document = new WordDocument();
//Opens an existing Word document into DocIO instance
document.Open(assembly.GetManifestResourceStream("Sample.Assets.Template.doc
x"), FormatType.Docx);
//Accesses the instance of the first section in the Word document
WSection section = document.Sections[0];
//Accesses the instance of the first table in the section
WTable table = section.Tables[0] as WTable;
//Resizes the table to fit the contents respect to the contents
table.AutoFit(AutoFitType.FitToContent);
//Accesses the instance of the second table in the section
table = section.Tables[1] as WTable;
//Resizes the table to fit the contents respect to window/page width
table.AutoFit(AutoFitType.FitToWindow);
//Accesses the instance of the third table in the section
table = section.Tables[2] as WTable;
//Resizes the table to fit the contents respect to fixed column width
table.AutoFit(AutoFitType.FixedColumnWidth);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document instance
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```
//Creates an instance of WordDocument class (Empty Word Document)
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument();
//Opens an existing Word document into DocIO instance
document.Open(fileStreamPath, FormatType.Docx);
//Accesses the instance of the first section in the Word document
WSection section = document.Sections[0];
//Accesses the instance of the first table in the section
WTable table = section.Tables[0] as WTable;
//Resizes the table to fit the contents respect to the contents
table.AutoFit(AutoFitType.FitToContent);
//Accesses the instance of the second table in the section
table = section.Tables[1] as WTable;
//Resizes the table to fit the contents respect to window/page width
table.AutoFit(AutoFitType.FitToWindow);
//Accesses the instance of the third table in the section
table = section.Tables[2] as WTable;
//Resizes the table to fit the contents respect to fixed column width
table.AutoFit(AutoFitType.FixedColumnWidth);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Creates an instance of WordDocument class (Empty Word Document)
WordDocument document = new WordDocument();
//Opens an existing Word document into DocIO instance
document.Open(assembly.GetManifestResourceStream("Sample.Assets.Template.doc
x"), FormatType.Docx);
//Accesses the instance of the first section in the Word document
WSection section = document.Sections[0];
//Accesses the instance of the first table in the section
WTable table = section.Tables[0] as WTable;
//Resizes the table to fit the contents respect to the contents
table.AutoFit(AutoFitType.FitToContent);
//Accesses the instance of the second table in the section
table = section.Tables[1] as WTable;
//Resizes the table to fit the contents respect to window/page width
table.AutoFit(AutoFitType.FitToWindow);
//Accesses the instance of the third table in the section
table = section.Tables[2] as WTable;
//Resizes the table to fit the contents respect to fixed column width
table.AutoFit(AutoFitType.FixedColumnWidth);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
```



```
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Closes the document instance
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Note: In ASP.NET Core, UWP, and Xamarin platforms, to apply autofit for table in a Word document we recommend you to use Word to PDF [assemblies](#) or [NuGet](#) packages as a reference in your application.

Working with Table Style

A table style defines a set of table, row, cell and paragraph level formatting that can be applied to a table. `WTableStyle` instance represents table style in a Word document.

Note: Essential DocIO currently provides support for table styles in DOCX and WordML formats alone. The visual appearance is also preserved in Word to PDF, Word to Image, and Word to HTML conversions.

The following code example illustrates how to apply the built-in table styles to the table.

C#

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument("Table.docx", FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Applies "LightShading" built-in style to table
table.ApplyStyle(BuiltinTableStyle.LightShading);
//Saves and closes the document instance
document.Save("TableStyle.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates an instance of WordDocument class
Dim document As New WordDocument("Table.docx", FormatType.Docx)
Dim section As WSection = document.Sections(0)
Dim table As WTable = TryCast(section.Tables(0), WTable)
'Applies "LightShading" built-in style to table
table.ApplyStyle(BuiltinTableStyle.LightShading)
'Saves and closes the document instance
document.Save("TableStyle.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Table.docx"),
FormatType.Docx);
```

```

WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Applies "LightShading" built-in style to table
table.ApplyStyle(BuiltinTableStyle.LightShading);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "TableStyle.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class
FileStream filePath = new FileStream("Table.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(filePath, FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Applies "LightShading" built-in style to table
table.ApplyStyle(BuiltinTableStyle.LightShading);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "TableStyle.docx");

```

XAMARIN

```

//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Table.d
ocx"), FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Applies "LightShading" built-in style to table
table.ApplyStyle(BuiltinTableStyle.LightShading);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TableStyle.docx",
"application/msword", stream);
//Closes the document instance
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin

```

Table style options

Once you have applied a table style, you can enable or disable the special formatting of the table. There are six options: first column, last column, banded rows, banded columns, header row and last row.

The following code example illustrates how to enable and disable the special table formatting options of the table styles

C#

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument("Table.docx", FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Applies "LightShading" built-in style to table
table.ApplyStyle(BuiltinTableStyle.LightShading);
//Enables special formatting for banded columns of the table
table.ApplyStyleForBandedColumns = true;
//Enables special formatting for banded rows of the table
table.ApplyStyleForBandedRows = true;
//Disables special formatting for first column of the table
table.ApplyStyleForFirstColumn = false;
//Enables special formatting for header row of the table
table.ApplyStyleForHeaderRow = true;
//Enables special formatting for last column of the table
table.ApplyStyleForLastColumn = true;
//Disables special formatting for last row of the table
table.ApplyStyleForLastRow = false;
//Saves and closes the document instance
document.Save("TableStyle.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates an instance of WordDocument class
Dim document As New WordDocument("Table.docx", FormatType.Docx)
Dim section As WSection = document.Sections(0)
Dim table As WTable = TryCast(section.Tables(0), WTable)
'Applies "LightShading" built-in style to table
table.ApplyStyle(BuiltinTableStyle.LightShading)
'Enables special formatting for banded columns of the table
table.ApplyStyleForBandedColumns = True
'Enables special formatting for banded rows of the table
table.ApplyStyleForBandedRows = True
'Disables special formatting for first column of the table
table.ApplyStyleForFirstColumn = False
'Enables special formatting for header row of the table
table.ApplyStyleForHeaderRow = True
'Enables special formatting for last column of the table
table.ApplyStyleForLastColumn = True
'Disables special formatting for last row of the table
table.ApplyStyleForLastRow = False
'Saves and closes the document instance
document.Save("TableStyle.docx", FormatType.Docx)
document.Close()
```

UWP

```

//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Table.docx"),
FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Applies "LightShading" built-in style to table
table.ApplyStyle(BuiltInTableStyle.LightShading);
//Enables special formatting for banded columns of the table
table.ApplyStyleForBandedColumns = true;
//Enables special formatting for banded rows of the table
table.ApplyStyleForBandedRows = true;
//Disables special formatting for first column of the table
table.ApplyStyleForFirstColumn = false;
//Enables special formatting for header row of the table
table.ApplyStyleForHeaderRow = true;
//Enables special formatting for last column of the table
table.ApplyStyleForLastColumn = true;
//Disables special formatting for last row of the table
table.ApplyStyleForLastRow = false;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "TableStyle.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class
FileStream fileStreamPath = new FileStream("Table.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Applies "LightShading" built-in style to table
table.ApplyStyle(BuiltInTableStyle.LightShading);
//Enables special formatting for banded columns of the table
table.ApplyStyleForBandedColumns = true;
//Enables special formatting for banded rows of the table
table.ApplyStyleForBandedRows = true;
//Disables special formatting for first column of the table
table.ApplyStyleForFirstColumn = false;
//Enables special formatting for header row of the table
table.ApplyStyleForHeaderRow = true;
//Enables special formatting for last column of the table
table.ApplyStyleForLastColumn = true;
//Disables special formatting for last row of the table
table.ApplyStyleForLastRow = false;

```

```
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "TableStyle.docx");
```

XAMARIN

```
//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Table.d
ocx"), FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Applies "LightShading" built-in style to table
table.ApplyStyle(BuiltInTableStyle.LightShading);
//Enables special formatting for banded columns of the table
table.ApplyStyleForBandedColumns = true;
//Enables special formatting for banded rows of the table
table.ApplyStyleForBandedRows = true;
//Disables special formatting for first column of the table
table.ApplyStyleForFirstColumn = false;
//Enables special formatting for header row of the table
table.ApplyStyleForHeaderRow = true;
//Enables special formatting for last column of the table
table.ApplyStyleForLastColumn = true;
//Disables special formatting for last row of the table
table.ApplyStyleForLastRow = false;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TableStyle.docx",
"application/msword", stream);
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Custom table style

The following code example illustrates how to apply a custom table style to table.

C#

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument("Table.docx", FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Adds a new custom table style
WTableStyle tableStyle = document.AddTableStyle("CustomStyle") as
WTableStyle;
```

```

//Applies formatting for whole table
tableStyle.TableProperties.RowStripe = 1;
tableStyle.TableProperties.ColumnStripe = 1;
tableStyle.TableProperties.Paddings.Top = 0;
tableStyle.TableProperties.Paddings.Bottom = 0;
tableStyle.TableProperties.Paddings.Left = 5.4f;
tableStyle.TableProperties.Paddings.Right = 5.4f;
//Applies conditional formatting for first row
ConditionalFormattingStyle firstRowStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.FirstRow);
firstRowStyle.CharacterFormat.Bold = true;
firstRowStyle.CharacterFormat.TextColor = Color.FromArgb(255, 255, 255, 255);
firstRowStyle.CellProperties.BackColor = Color.Blue;
//Applies conditional formatting for first column
ConditionalFormattingStyle firstColumnStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.FirstColumn);
firstColumnStyle.CharacterFormat.Bold = true;
//Applies conditional formatting for odd row
ConditionalFormattingStyle oddRowBandingStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.OddRowBanding);
oddRowBandingStyle.CellProperties.BackColor = Color.WhiteSmoke;
//Applies the custom table style to the table
table.ApplyStyle("CustomStyle");
//Saves and closes the document instance
document.Save("TableStyle.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates an instance of WordDocument class
Dim document As New WordDocument("Table.docx", FormatType.Docx)
Dim section As WSection = document.Sections(0)
Dim table As WTable = TryCast(section.Tables(0), WTable)
'Adds a new custom table style
Dim tableStyle As WTableStyle =
TryCast(document.AddTableStyle("CustomStyle"), WTableStyle)
'Applies formatting for whole table
tableStyle.TableProperties.RowStripe = 1
tableStyle.TableProperties.ColumnStripe = 1
tableStyle.TableProperties.Paddings.Top = 0
tableStyle.TableProperties.Paddings.Bottom = 0
tableStyle.TableProperties.Paddings.Left = 5.4F
tableStyle.TableProperties.Paddings.Right = 5.4F
'Applies conditional formatting for first row
Dim firstRowStyle As ConditionalFormattingStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.FirstRow)
firstRowStyle.CharacterFormat.Bold = True
firstRowStyle.CharacterFormat.TextColor = Color.FromArgb(255, 255, 255, 255)
firstRowStyle.CellProperties.BackColor = Color.Blue
'Applies conditional formatting for first column

```

```

Dim firstColumnStyle As ConditionalFormattingStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.FirstCo
lumn)
firstColumnStyle.CharacterFormat.Bold = True
'Applies conditional formatting for odd row
Dim oddRowBandingStyle As ConditionalFormattingStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.OddRowB
anding)
oddRowBandingStyle.CellProperties.BackgroundColor = Color.WhiteSmoke
'Applies the custom table style to the table
table.ApplyStyle("CustomStyle")
'Saves and closes the document instance
document.Save("TableStyle.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Table.docx"),
FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Adds a new custom table style
WTableStyle tableStyle = document.AddTableStyle("CustomStyle") as
WTableStyle;
//Applies formatting for whole table
tableStyle.TableProperties.RowStripe = 1;
tableStyle.TableProperties.ColumnStripe = 1;
tableStyle.TableProperties.Paddings.Top = 0;
tableStyle.TableProperties.Paddings.Bottom = 0;
tableStyle.TableProperties.Paddings.Left = 5.4f;
tableStyle.TableProperties.Paddings.Right = 5.4f;
//Applies conditional formatting for first row
ConditionalFormattingStyle firstRowStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.FirstRo
w);
firstRowStyle.CharacterFormat.Bold = true;
firstRowStyle.CharacterFormat.TextColor = Color.FromArgb(255, 255, 255,
255);
firstRowStyle.CellProperties.BackgroundColor = Color.Blue;
//Applies conditional formatting for first column
ConditionalFormattingStyle firstColumnStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.FirstCo
lumn);
firstColumnStyle.CharacterFormat.Bold = true;
//Applies conditional formatting for odd row
ConditionalFormattingStyle oddRowBandingStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.OddRowB
anding);
oddRowBandingStyle.CellProperties.BackgroundColor = Color.WhiteSmoke;
//Applies the custom table style to the table
table.ApplyStyle("CustomStyle");
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();

```

```
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "TableStyle.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates an instance of WordDocument class
FileStream fileStreamPath = new FileStream("Table.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Adds a new custom table style
WTableStyle tableStyle = document.AddTableStyle("CustomStyle") as
WTableStyle;
//Applies formatting for whole table
tableStyle.TableProperties.RowStripe = 1;
tableStyle.TableProperties.ColumnStripe = 1;
tableStyle.TableProperties.Paddings.Top = 0;
tableStyle.TableProperties.Paddings.Bottom = 0;
tableStyle.TableProperties.Paddings.Left = 5.4f;
tableStyle.TableProperties.Paddings.Right = 5.4f;
//Applies conditional formatting for first row
ConditionalFormattingStyle firstRowStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.FirstRow);
firstRowStyle.CharacterFormat.Bold = true;
firstRowStyle.CharacterFormat.TextColor = Color.FromArgb(255, 255, 255,
255);
firstRowStyle.CellProperties.BackColor = Color.Blue;
//Applies conditional formatting for first column
ConditionalFormattingStyle firstColumnStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.FirstColumn);
firstColumnStyle.CharacterFormat.Bold = true;
//Applies conditional formatting for odd row
ConditionalFormattingStyle oddRowBandingStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.OddRowBanding);
oddRowBandingStyle.CellProperties.BackColor = Color.WhiteSmoke;
//Applies the custom table style to the table
table.ApplyStyle("CustomStyle");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "TableStyle.docx");
```


XAMARIN

```

//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Table.docx"), FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Adds a new custom table style
WTableStyle tableStyle = document.AddTableStyle("CustomStyle") as
WTableStyle;
//Applies formatting for whole table
tableStyle.TableProperties.RowStripe = 1;
tableStyle.TableProperties.ColumnStripe = 1;
tableStyle.TableProperties.Paddings.Top = 0;
tableStyle.TableProperties.Paddings.Bottom = 0;
tableStyle.TableProperties.Paddings.Left = 5.4f;
tableStyle.TableProperties.Paddings.Right = 5.4f;
//Applies conditional formatting for first row
ConditionalFormattingStyle firstRowStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.FirstRow);
firstRowStyle.CharacterFormat.Bold = true;
firstRowStyle.CharacterFormat.TextColor = Color.FromArgb(255, 255, 255, 255);
firstRowStyle.CellProperties.BackgroundColor = Color.Blue;
//Applies conditional formatting for first column
ConditionalFormattingStyle firstColumnStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.FirstColumn);
firstColumnStyle.CharacterFormat.Bold = true;
//Applies conditional formatting for odd row
ConditionalFormattingStyle oddRowBandingStyle =
tableStyle.ConditionalFormattingStyles.Add(ConditionalFormattingType.OddRowBanding);
oddRowBandingStyle.CellProperties.BackgroundColor = Color.WhiteSmoke;
//Applies the custom table style to the table
table.ApplyStyle("CustomStyle");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TableStyle.docx",
"application/msword", stream);
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin

```

Merging cells vertically and horizontally

You can combine two or more table cells located in the same row or column into a single cell.

The following code example illustrates how to apply horizontal merge to specified range of cells in a specified row.

C#

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Horizontal merging of Table cells");
IWTable table = section.AddTable();
table.ResetCells(5, 5);
//Specifies the horizontal merge from second cell to fifth cell in third row
table.ApplyHorizontalMerge(2, 1, 4);
//Saves and closes the document instance
document.Save("HorizontalMerge.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates an instance of WordDocument class
Dim document As New WordDocument()
Dim section As IWSection = document.AddSection()
section.AddParagraph().AppendText("Horizontal merging of Table cells")
Dim table As IWTable = section.AddTable()
table.ResetCells(5, 5)
'Specifies the horizontal merge from second cell to fifth cell in third row
table.ApplyHorizontalMerge(2, 1, 4)
'Saves and closes the document instance
document.Save("HorizontalMerge.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Horizontal merging of Table cells");
IWTable table = section.AddTable();
table.ResetCells(5, 5);
//Specifies the horizontal merge from second cell to fifth cell in third row
table.ApplyHorizontalMerge(2, 1, 4);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "HorizontalMerge.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Horizontal merging of Table cells");
IWTable table = section.AddTable();
```

```

table.ResetCells(5, 5);
//Specifies the horizontal merge from second cell to fifth cell in third row
table.ApplyHorizontalMerge(2, 1, 4);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "HorizontalMerge.docx");

```

XAMARIN

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Horizontal merging of Table cells");
IWTable table = section.AddTable();
table.ResetCells(5, 5);
//Specifies the horizontal merge from second cell to fifth cell in third row
table.ApplyHorizontalMerge(2, 1, 4);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("HorizontalMerge.docx", "application/msword", stream);
document.Close();
//Please download the helper files from the below link to save the stream as file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin

```

The following code example illustrates how to apply vertical merge to specified range of rows in a specified column.

C#

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Vertical merging of Table cells");
IWTable table = section.AddTable();
table.ResetCells(5, 5);
//Specifies the vertical merge to the third cell, from second row to fifth row
table.ApplyVerticalMerge(2, 1, 4);
//Saves and closes the document instance
document.Save("VerticalMerge.docx", FormatType.Docx);
document.Close();

```

VB.NET

```
'Creates an instance of WordDocument class
```

```

Dim document As New WordDocument()
Dim section As IWSection = document.AddSection()
section.AddParagraph().AppendText("Vertical merging of Table cells")
Dim table As ITable = section.AddTable()
table.ResetCells(5, 5)
'Specifies the vertical merge to the third cell, from second row to fifth row
table.ApplyVerticalMerge(2, 1, 4)
'Saves and closes the document instance
document.Save("VerticalMerge.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Vertical merging of Table cells");
ITable table = section.AddTable();
table.ResetCells(5, 5);
//Specifies the vertical merge to the third cell, from second row to fifth row
table.ApplyVerticalMerge(2, 1, 4);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "VerticalMerge.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Vertical merging of Table cells");
ITable table = section.AddTable();
table.ResetCells(5, 5);
// Specifies the vertical merge to the third cell, from second row to fifth row
table.ApplyVerticalMerge(2, 1, 4);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "VerticalMerge.docx");

```

XAMARIN

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Vertical merging of Table cells");
IWTable table = section.AddTable();
table.ResetCells(5, 5);
//Specifies the vertical merge to the third cell, from second row to fifth row
table.ApplyVerticalMerge(2, 1, 4);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("VerticalMerge.docx", "application/msword", stream);
document.Close();
//Please download the helper files from the below link to save the stream as file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
```

The following code example illustrate how to create a table that contains horizontal merged cells.

C#

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Horizontal merging of Table cells");
IWTable table = section.AddTable();
table.ResetCells(2, 2);
//Adds content to table cell
table[0, 0].AddParagraph().AppendText("First row, First cell");
table[0, 1].AddParagraph().AppendText("First row, Second cell");
table[1, 0].AddParagraph().AppendText("Second row, First cell");
table[1, 1].AddParagraph().AppendText("Second row, Second cell");
//Specifies the horizontal merge start to first row, first cell
table[0, 0].CellFormat.HorizontalMerge = CellMerge.Start;
//Modifies the cell content
table[0, 0].Paragraphs[0].Text = "Horizontally merged cell";
//Specifies the horizontal merge continue to second row second cell
table[0, 1].CellFormat.HorizontalMerge = CellMerge.Continue;
//Saves and closes the document instance
document.Save("HorizontalMerge.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates an instance of WordDocument class
Dim document As New WordDocument()
Dim section As IWSection = document.AddSection()
section.AddParagraph().AppendText("Horizontal merging of Table cells")
Dim table As IWTable = section.AddTable()
table.ResetCells(2, 2)
'Adds content to table cell
table(0, 0).AddParagraph().AppendText("First row, First cell")
```

```

table(0, 1).AddParagraph().AppendText("First row, Second cell")
table(1, 0).AddParagraph().AppendText("Second row, First cell")
table(1, 1).AddParagraph().AppendText("Second row, Second cell")
'Specifies the horizontal merge start to first row, first cell
table(0, 0).CellFormat.HorizontalMerge = CellMerge.Start
'Modifies the cell content
table(0, 0).Paragraphs(0).Text = "Horizontally merged cell"
'Specifies the horizontal merge continue to second row second cell
table(0, 1).CellFormat.HorizontalMerge = CellMerge.[Continue]
'Saves and closes the document instance
document.Save("HorizontalMerge.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Horizontal merging of Table cells");
IWTable table = section.AddTable();
table.ResetCells(2, 2);
//Adds content to table cell
table[0, 0].AddParagraph().AppendText("First row, First cell");
table[0, 1].AddParagraph().AppendText("First row, Second cell");
table[1, 0].AddParagraph().AppendText("Second row, First cell");
table[1, 1].AddParagraph().AppendText("Second row, Second cell");
//Specifies the horizontal merge start to first row, first cell
table[0, 0].CellFormat.HorizontalMerge = CellMerge.Start;
//Modifies the cell content
table[0, 0].Paragraphs[0].Text = "Horizontally merged cell";
//Specifies the horizontal merge continue to second row second cell
table[0, 1].CellFormat.HorizontalMerge = CellMerge.Continue;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "HorizontalMerge.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Horizontal merging of Table cells");
IWTable table = section.AddTable();
table.ResetCells(2, 2);
//Adds content to table cell
table[0, 0].AddParagraph().AppendText("First row, First cell");
table[0, 1].AddParagraph().AppendText("First row, Second cell");
table[1, 0].AddParagraph().AppendText("Second row, First cell");
table[1, 1].AddParagraph().AppendText("Second row, Second cell");
//Specifies the horizontal merge start to first row, first cell

```

```

table[0, 0].CellFormat.HorizontalMerge = CellMerge.Start;
//Modifies the cell content
table[0, 0].Paragraphs[0].Text = "Horizontally merged cell";
//Specifies the horizontal merge continue to second row second cell
table[0, 1].CellFormat.HorizontalMerge = CellMerge.Continue;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "HorizontalMerge.docx");

```

XAMARIN

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Horizontal merging of Table cells");
IWTable table = section.AddTable();
table.ResetCells(2, 2);
//Adds content to table cell
table[0, 0].AddParagraph().AppendText("First row, First cell");
table[0, 1].AddParagraph().AppendText("First row, Second cell");
table[1, 0].AddParagraph().AppendText("Second row, First cell");
table[1, 1].AddParagraph().AppendText("Second row, Second cell");
//Specifies the horizontal merge start to first row, first cell
table[0, 0].CellFormat.HorizontalMerge = CellMerge.Start;
//Modifies the cell content
table[0, 0].Paragraphs[0].Text = "Horizontally merged cell";
//Specifies the horizontal merge continue to second row second cell
table[0, 1].CellFormat.HorizontalMerge = CellMerge.Continue;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("HorizontalMerge.do
cx", "application/msword", stream);
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example illustrates how to create a table with vertical merged cells.

C#

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Vertical merging of Table cells");
IWTable table = section.AddTable();
table.ResetCells(2, 2);
//Adds content to table cells

```

```

table[0, 0].AddParagraph().AppendText("First row, First cell");
table[0, 1].AddParagraph().AppendText("First row, Second cell");
table[1, 0].AddParagraph().AppendText("Second row, First cell");
table[1, 1].AddParagraph().AppendText("Second row, Second cell");
//Specifies the vertical merge start to first row first cell
table[0, 0].CellFormat.VerticalMerge = CellMerge.Start;
//Modifies the cell content
table[0, 0].Paragraphs[0].Text = "Vertically merged cell";
//Specifies the vertical merge continue to second row first cell
table[1, 0].CellFormat.VerticalMerge = CellMerge.Continue;
//Saves and closes the document instance
document.Save("VerticalMerge.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates an instance of WordDocument class
Dim document As New WordDocument()
Dim section As IWSection = document.AddSection()
section.AddParagraph().AppendText("Vertical merging of Table cells")
Dim table As ITable = section.AddTable()
table.ResetCells(2, 2)
'Adds content to table cells
table(0, 0).AddParagraph().AppendText("First row, First cell")
table(0, 1).AddParagraph().AppendText("First row, Second cell")
table(1, 0).AddParagraph().AppendText("Second row, First cell")
table(1, 1).AddParagraph().AppendText("Second row, Second cell")
'Specifies the vertical merge start to first row first cell
table(0, 0).CellFormat.VerticalMerge = CellMerge.Start
'Modifies the cell content
table(0, 0).Paragraphs(0).Text = "Vertically merged cell"
'Specifies the vertical merge continue to second row first cell
table(1, 0).CellFormat.VerticalMerge = CellMerge.Continue
'Saves and closes the document instance
document.Save("VerticalMerge.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Vertical merging of Table cells");
ITable table = section.AddTable();
table.ResetCells(2, 2);
//Adds content to table cells
table[0, 0].AddParagraph().AppendText("First row, First cell");
table[0, 1].AddParagraph().AppendText("First row, Second cell");
table[1, 0].AddParagraph().AppendText("Second row, First cell");
table[1, 1].AddParagraph().AppendText("Second row, Second cell");
//Specifies the vertical merge start to first row first cell
table[0, 0].CellFormat.VerticalMerge = CellMerge.Start;
//Modifies the cell content
table[0, 0].Paragraphs[0].Text = "Vertically merged cell";
//Specifies the vertical merge continue to second row first cell
table[1, 0].CellFormat.VerticalMerge = CellMerge.Continue;

```



```
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "VerticalMerge.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Vertical merging of Table cells");
IWTable table = section.AddTable();
table.ResetCells(2, 2);
//Adds content to table cells
table[0, 0].AddParagraph().AppendText("First row, First cell");
table[0, 1].AddParagraph().AppendText("First row, Second cell");
table[1, 0].AddParagraph().AppendText("Second row, First cell");
table[1, 1].AddParagraph().AppendText("Second row, Second cell");
//Specifies the vertical merge start to first row first cell
table[0, 0].CellFormat.VerticalMerge = CellMerge.Start;
//Modifies the cell content
table[0, 0].Paragraphs[0].Text = "Vertically merged cell";
//Specifies the vertical merge continue to second row first cell
table[1, 0].CellFormat.VerticalMerge = CellMerge.Continue;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "VerticalMerge.docx");
```

XAMARIN

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
section.AddParagraph().AppendText("Vertical merging of Table cells");
IWTable table = section.AddTable();
table.ResetCells(2, 2);
//Adds content to table cells
table[0, 0].AddParagraph().AppendText("First row, First cell");
table[0, 1].AddParagraph().AppendText("First row, Second cell");
table[1, 0].AddParagraph().AppendText("Second row, First cell");
table[1, 1].AddParagraph().AppendText("Second row, Second cell");
//Specifies the vertical merge start to first row first cell
table[0, 0].CellFormat.VerticalMerge = CellMerge.Start;
//Modifies the cell content
table[0, 0].Paragraphs[0].Text = "Vertically merged cell";
//Specifies the vertical merge continue to second row first cell
```

```

table[1, 0].CellFormat.VerticalMerge = CellMerge.Continue;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("VerticalMerge.docx", "application/msword", stream);
document.Close();
//Please download the helper files from the below link to save the stream as file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin

```

Specifying table header row to repeat on each page

You can specify one or more rows in a table to be repeated as header row at the top of each page, when the table spans across multiple pages.

- In the case of a single header row, it must be the first row in the table.
- In the case of multiple header rows, then header rows must be consecutive from the first row of the table.

Note: Heading rows do not have any effect with nested tables in Microsoft Word as well as DocIO

The following code example illustrates how to create a table with a single header row.

C#

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
IWTable table = section.AddTable();
table.ResetCells(50, 1);
WTableRow row = table.Rows[0];
//Specifies the first row as a header row of the table
row.IsHeader = true;
row.Height = 20;
row.HeightType = TableRowHeightType.AtLeast;
row.Cells[0].AddParagraph().AppendText("Header Row");
for (int i = 1; i < 50; i++)
{
    row = table.Rows[i];
    row.Height = 20;
    row.HeightType = TableRowHeightType.AtLeast;
    row.Cells[0].AddParagraph().AppendText("Text in Row" + i.ToString());
}
//Saves and closes the document instance
document.Save("TableWithHeaderRow.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates an instance of WordDocument class
Dim document As New WordDocument()
Dim section As IWSection = document.AddSection()

```

```

Dim table As ITable = section.AddTable()
table.ResetCells(50, 1)
Dim row As WTableRow = table.Rows(0)
'Specifies the first row as a header row of the table
row.IsHeader = True
row.Height = 20
row.HeightType = TableRowHeightType.AtLeast
row.Cells(0).AddParagraph().AppendText("Header Row")
For i As Integer = 1 To 49
row = table.Rows(i)
row.Height = 20
row.HeightType = TableRowHeightType.AtLeast
row.Cells(0).AddParagraph().AppendText("Text in Row" + i.ToString())
Next
'Saves and closes the document instance
document.Save("TableWithHeaderRow.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
ITable table = section.AddTable();
table.ResetCells(50, 1);
WTableRow row = table.Rows[0];
//Specifies the first row as a header row of the table
row.IsHeader = true;
row.Height = 20;
row.HeightType = TableRowHeightType.AtLeast;
row.Cells[0].AddParagraph().AppendText("Header Row");
for (int i = 1; i < 50; i++)
{
row = table.Rows[i];
row.Height = 20;
row.HeightType = TableRowHeightType.AtLeast;
row.Cells[0].AddParagraph().AppendText("Text in Row" + i.ToString());
}
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "TableWithHeaderRow.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
ITable table = section.AddTable();
table.ResetCells(50, 1);
WTableRow row = table.Rows[0];

```

```
//Specifies the first row as a header row of the table
row.IsHeader = true;
row.Height = 20;
row.HeightType = TableRowHeightType.AtLeast;
row.Cells[0].AddParagraph().AppendText("Header Row");
for (int i = 1; i < 50; i++)
{
    row = table.Rows[i];
    row.Height = 20;
    row.HeightType = TableRowHeightType.AtLeast;
    row.Cells[0].AddParagraph().AppendText("Text in Row" + i.ToString());
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "TableWithHeaderRow.docx");
```

XAMARIN

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
IWTable table = section.AddTable();
table.ResetCells(50, 1);
WTableRow row = table.Rows[0];
//Specifies the first row as a header row of the table
row.IsHeader = true;
row.Height = 20;
row.HeightType = TableRowHeightType.AtLeast;
row.Cells[0].AddParagraph().AppendText("Header Row");
for (int i = 1; i < 50; i++)
{
    row = table.Rows[i];
    row.Height = 20;
    row.HeightType = TableRowHeightType.AtLeast;
    row.Cells[0].AddParagraph().AppendText("Text in Row" + i.ToString());
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TableWithHeaderRow.docx", "application/msword", stream);
document.Close();
//Please download the helper files from the below link to save the stream as file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
```

Keeping rows from breaking across pages

You can enable or disable the table row content to split across multiple pages, when the row contents do not fit in a previous page.

The following code example illustrates how to disable all the table rows from splitting across multiple pages.

C#

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument("Template.docx");
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Disables breaking across pages for all rows in the table.
foreach (WTableRow row in table.Rows)
    row.RowFormat.IsBreakAcrossPages = false;
//Saves and closes the document instance
document.Save("Result.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates an instance of WordDocument class
Dim document As New WordDocument("Template.docx")
Dim section As WSection = document.Sections(0)
Dim table As WTable = TryCast(section.Tables(0), WTable)
'Disables breaking across pages for all rows in the table.
For Each row As WTableRow In table.Rows
    row.RowFormat.IsBreakAcrossPages = False
Next
'Saves and closes the document instance
document.Save("Result.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx"), FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Disables breaking across pages for all rows in the table.
foreach (WTableRow row in table.Rows)
    row.RowFormat.IsBreakAcrossPages = false;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates an instance of WordDocument class
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Disables breaking across pages for all rows in the table.
foreach (WTableRow row in table.Rows)
row.RowFormat.IsBreakAcrossPages = false;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Templat
e.docx"), FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Disables breaking across pages for all rows in the table.
foreach (WTableRow row in table.Rows)
row.RowFormat.IsBreakAcrossPages = false;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Iterating through table elements

The following code example illustrates how to iterate through the table and apply back color to a particular cell.

C#

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument("Template.docx");
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Iterates the rows of the table
foreach (WTableRow row in table.Rows)
```

```

{
    //Iterates through the cells of rows
    foreach (WTableCell cell in row.Cells)
    {
        //Iterates through the paragraphs of the cell
        foreach (WParagraph paragraph in cell.Paragraphs)
        {
            //When the paragraph contains text Panda then apply green as back color to cell
            if (paragraph.Text.Contains("panda"))
            {
                cell.CellFormat.BackColor = Color.Green;
            }
        }
    }
    //Saves and closes the document instance
    document.Save("Sample.docx", FormatType.Docx);
    document.Close();
}

```

VB.NET

```

'Creates an instance of WordDocument class
Dim document As New WordDocument("Template.docx")
Dim section As WSection = document.Sections(0)
Dim table As WTable = TryCast(section.Tables(0), WTable)
'Iterates the rows of the table
For Each row As WTableRow In table.Rows
    'Iterates through the cells of rows
    For Each cell As WTableCell In row.Cells
        'Iterates through the paragraphs of the cell
        For Each paragraph As WParagraph In cell.Paragraphs
            'When the paragraph contains text Panda then apply green as back color to cell
            If paragraph.Text.Contains("panda") Then
                cell.CellFormat.BackColor = Color.Green
            End If
        Next
    Next
Next
'Saves and closes the document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx"),
FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Iterates the rows of the table
foreach (WTableRow row in table.Rows)
{
    //Iterates through the cells of rows
    foreach (WTableCell cell in row.Cells)

```

```

{
    //Iterates through the paragraphs of the cell
    foreach (WParagraph paragraph in cell.Paragraphs)
    {
        //When the paragraph contains text Panda then apply green as back color to cell
        if (paragraph.Text.Contains("panda"))
            cell.CellFormat.BackColor = Color.Green;
    }
}

//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Iterates the rows of the table
foreach (WTableRow row in table.Rows)
{
    //Iterates through the cells of rows
    foreach (WTableCell cell in row.Cells)
    {
        //Iterates through the paragraphs of the cell
        foreach (WParagraph paragraph in cell.Paragraphs)
        {
            //When the paragraph contains text Panda then apply green as back color to cell
            if (paragraph.Text.Contains("panda"))
                cell.CellFormat.BackColor = Color.Green;
        }
    }
}

//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN


```

//Creates an instance of WordDocument class
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Assets.Template.docx"), FormatType.Docx);
WSection section = document.Sections[0];
WTable table = section.Tables[0] as WTable;
//Iterates the rows of the table
foreach (WTableRow row in table.Rows)
{
    //Iterates through the cells of rows
    foreach (WTableCell cell in row.Cells)
    {
        //Iterates through the paragraphs of the cell
        foreach (WParagraph paragraph in cell.Paragraphs)
        {
            //When the paragraph contains text Panda then apply green as back color to cell
            if (paragraph.Text.Contains("panda"))
            cell.CellFormat.BackColor = Color.Green;
        }
    }
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
document.Close();
//Please download the helper files from the below link to save the stream as file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin

```

Working with Bookmarks

A bookmark identifies a location or a selection of text within a document that you can name and identify for future reference.

In Essential DocIO, bookmark is represented by Bookmark instance that is a pair of BookmarkStart and BookmarkEnd. BookmarkStart represents start point of a bookmark and BookmarkEnd represents end point of a bookmark. Every Word document contains a collection of bookmarks that are accessible through the Bookmarks property of WordDocument class.

Adding a bookmark

The following code example shows how to add a bookmark in Word document.

C#

```

//Creates an instance of WordDocument class (Empty Word Document)
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Northwind Database");

```

```

paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
//Adds a paragraph into section
paragraph = section.AddParagraph();
//Adds a new bookmark start into paragraph with name "Northwind"
paragraph.AppendBookmarkStart("Northwind");
//Adds a text between the bookmark start and end into paragraph
paragraph.AppendText("The Northwind sample database (Northwind.mdb) is
included with all versions of Access. It provides data you can experiment
with and database objects that demonstrate features you might want to
implement in your own databases.");
//Adds a new bookmark end into paragraph with name " Northwind "
paragraph.AppendBookmarkEnd("Northwind");
//Adds a text after the bookmark end
paragraph.AppendText(" Using Northwind, you can become familiar with how a
relational database is structured and how the database objects work together
to help you enter, store, manipulate, and print your data.");
//Saves the document in the given name and format
document.Save("Bookmarks.docx", FormatType.Docx);
//Releases the resources occupied by WordDocument instance
document.Close();

```

VB.NET

```

'Creates an instance of WordDocument class (Empty Word Document)
Dim document As New WordDocument()
'Adds a new section into the Word Document
Dim section As IWSection = document.AddSection()
'Adds a new paragraph into Word document and appends text into paragraph
Dim paragraph As IWParagraph = section.AddParagraph()
paragraph.AppendText("Northwind Database")
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center
'Adds a paragraph into section
paragraph = section.AddParagraph()
'Adds a new bookmark start into paragraph with name "Northwind"
paragraph.AppendBookmarkStart("Northwind")
'Adds a text between the bookmark start and end into paragraph
paragraph.AppendText("The Northwind sample database (Northwind.mdb) is
included with all versions of Access. It provides data you can experiment
with and database objects that demonstrate features you might want to
implement in your own databases.")
'Adds a new bookmark end into paragraph with name " Northwind "
paragraph.AppendBookmarkEnd("Northwind")
'Adds a text after the bookmark end
paragraph.AppendText(" Using Northwind, you can become familiar with how a
relational database is structured and how the database objects work together
to help you enter, store, manipulate, and print your data.")
'Saves the document in the given name and format
document.Save("Bookmarks.docx", FormatType.Docx)
'Releases the resources occupied by WordDocument instance
document.Close()

```

UWP

```

//Creates an instance of WordDocument class (Empty Word Document)

```

```

WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Northwind Database");
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
//Adds a paragraph into section
paragraph = section.AddParagraph();
//Adds a new bookmark start into paragraph with name "Northwind"
paragraph.AppendBookmarkStart("Northwind");
//Adds a text between the bookmark start and end into paragraph
paragraph.AppendText("The Northwind sample database (Northwind.mdb) is
included with all versions of Access. It provides data you can experiment
with and database objects that demonstrate features you might want to
implement in your own databases.");
//Adds a new bookmark end into paragraph with name " Northwind "
paragraph.AppendBookmarkEnd("Northwind");
//Adds a text after the bookmark end
paragraph.AppendText(" Using Northwind, you can become familiar with how a
relational database is structured and how the database objects work together
to help you enter, store, manipulate, and print your data.");
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Bookmarks.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class (Empty Word Document)
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Northwind Database");
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
//Adds a paragraph into section
paragraph = section.AddParagraph();
//Adds a new bookmark start into paragraph with name "Northwind"
paragraph.AppendBookmarkStart("Northwind");
//Adds a text between the bookmark start and end into paragraph
paragraph.AppendText("The Northwind sample database (Northwind.mdb) is
included with all versions of Access. It provides data you can experiment
with and database objects that demonstrate features you might want to
implement in your own databases.");
//Adds a new bookmark end into paragraph with name " Northwind "
paragraph.AppendBookmarkEnd("Northwind");

```

```
//Adds a text after the bookmark end
paragraph.AppendText(" Using Northwind, you can become familiar with how a
relational database is structured and how the database objects work together
to help you enter, store, manipulate, and print your data.");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Bookmarks.docx");
```

XAMARIN

```
//Creates an instance of WordDocument class (Empty Word Document)
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Northwind Database");
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
//Adds a paragraph into section
paragraph = section.AddParagraph();
//Adds a new bookmark start into paragraph with name "Northwind"
paragraph.AppendBookmarkStart("Northwind");
//Adds a text between the bookmark start and end into paragraph
paragraph.AppendText("The Northwind sample database (Northwind.mdb) is
included with all versions of Access. It provides data you can experiment
with and database objects that demonstrate features you might want to
implement in your own databases.");
//Adds a new bookmark end into paragraph with name " Northwind "
paragraph.AppendBookmarkEnd("Northwind");
//Adds a text after the bookmark end
paragraph.AppendText(" Using Northwind, you can become familiar with how a
relational database is structured and how the database objects work together
to help you enter, store, manipulate, and print your data.");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Bookmarks.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Obtaining a bookmark instance

The following code example shows how to retrieve an instance of bookmark from a Word document.

C#

```
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument("Bookmarks.docx", FormatType.Docx);
//Gets the bookmark instance by using FindByName method of
BookmarkCollection with bookmark name
Syncfusion.DocIO.DLS.Bookmark bookmark =
document.Bookmarks.FindByName("Northwind");
//Accesses the bookmark start's owner paragraph by using bookmark and
changes its back color
bookmark.BookmarkStart.OwnerParagraph.ParagraphFormat.BackColor =
Color.AliceBlue;
document.Save("Result.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Loads an existing Word document into DocIO instance
Dim document As New WordDocument("Bookmarks.docx", FormatType.Docx)
'Gets the bookmark instance by using FindByName method of BookmarkCollection
with bookmark name
Dim bookmark As Syncfusion.DocIO.DLS.Bookmark =
document.Bookmarks.FindByName("Northwind")
'Accesses the bookmark start's owner paragraph by using bookmark and changes
its back color
bookmark.BookmarkStart.OwnerParagraph.ParagraphFormat.BackColor =
Color.AliceBlue
document.Save("Result.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new WordDocument("Sample.Assets.Bookmarks.docx",
FormatType.Docx);
//Gets the bookmark instance by using FindByName method of
BookmarkCollection with bookmark name
Syncfusion.DocIO.DLS.Bookmark bookmark =
document.Bookmarks.FindByName("Northwind");
//Accesses the bookmark start's owner paragraph by using bookmark and
changes its back color
bookmark.BookmarkStart.OwnerParagraph.ParagraphFormat.BackColor =
Color.AliceBlue;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Loads an existing Word document into DocIO instance
FileStream fileStreamPath = new FileStream(@"Bookmarks.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Gets the bookmark instance by using FindByName method of
BookmarkCollection with bookmark name
Syncfusion.DocIO.DLS.Bookmark bookmark =
document.Bookmarks.FindByName("Northwind");
//Accesses the bookmark start's owner paragraph by using bookmark and
changes its back color
bookmark.BookmarkStart.OwnerParagraph.ParagraphFormat.BackColor =
Color.AliceBlue;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Bookmar
ks.docx"), FormatType.Docx);
//Gets the bookmark instance by using FindByName method of
BookmarkCollection with bookmark name
Syncfusion.DocIO.DLS.Bookmark bookmark =
document.Bookmarks.FindByName("Northwind");
//Accesses the bookmark start's owner paragraph by using bookmark and
changes its back color
bookmark.BookmarkStart.OwnerParagraph.ParagraphFormat.BackColor =
Color.AliceBlue;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Removing a Bookmark from Word document

The following code example shows how to remove a bookmark from Word document.

C#

```
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument("Bookmarks.docx", FormatType.Docx);
//Gets the bookmark instance by using FindByName method of
BookmarkCollection with bookmark name
Bookmark bookmark = document.Bookmarks.FindByName("Northwind");
//Removes the bookmark named "Northwind" from Word document.
document.Bookmarks.Remove(bookmark);
document.Save("Result.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Loads an existing Word document into DocIO instance
Dim document As New WordDocument("Bookmarks.docx", FormatType.Docx)
'Gets the bookmark instance by using FindByName method of BookmarkCollection
with bookmark name
Dim bookmark As Bookmark = document.Bookmarks.FindByName("Northwind")
'Removes the bookmark named "Northwind" from Word document.
document.Bookmarks.Remove(bookmark)
document.Save("Result.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Bookmarks.doc
x"), FormatType.Docx);
//Gets the bookmark instance by using FindByName method of
BookmarkCollection with bookmark name
Bookmark bookmark = document.Bookmarks.FindByName("Northwind");
//Removes the bookmark named "Northwind" from Word document.
document.Bookmarks.Remove(bookmark);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Loads an existing Word document into DocIO instance
FileStream filePath = new FileStream(@"Bookmarks.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(filePath, FormatType.Docx);
//Gets the bookmark instance by using FindByName method of
BookmarkCollection with bookmark name
Bookmark bookmark = document.Bookmarks.FindByName("Northwind");
//Removes the bookmark named "Northwind" from Word document.
```

```
document.Bookmarks.Remove(bookmark);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Bookmarks.docx"), FormatType.Docx);
//Gets the bookmark instance by using FindByName method of
BookmarkCollection with bookmark name
Bookmark bookmark = document.Bookmarks.FindByName("Northwind");
//Removes the bookmark named "Northwind" from Word document.
document.Bookmarks.Remove(bookmark);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
```

Retrieving contents within a bookmark

BookmarkNavigator is used for navigating to a bookmark in a Word document. You can retrieve, replace and delete the content of a specified bookmark by using BookmarkNavigator.

You can get the content between bookmark start and bookmark end of the specified bookmark in two ways:

1. You can use `GetBookmarkContent` method for retrieving content as collection of body items when the bookmark start and bookmark end are preserved in a single section.
2. You can use `GetContent` method for retrieving content as collection of sections when the bookmark start and bookmark end are preserved in different sections.

The following code example shows how to retrieve the specified bookmark content by using `GetBookmarkContent` method in a Word document.

C#

```
WordDocument document = new WordDocument("Bookmarks.docx", FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
```



```

BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content
TextBodyPart part = bookmarkNavigator.GetBookmarkContent();
//Adds the retrieved content into another new section
document.AddSection();
for (int i = 0; i < part.BodyItems.Count; i++)
document.LastSection.Body.ChildEntities.Add(part.BodyItems[i]);
document.Save("Result.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

Dim document As New WordDocument("Bookmarks.docx", FormatType.Docx)
'Creates the bookmark navigator instance to access the bookmark
Dim bookmarkNavigator As New BookmarksNavigator(document)
'Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind")
'Gets the bookmark content
Dim part As TextBodyPart = bookmarkNavigator.GetBookmarkContent()
'Adds the retrieved content into another new section
document.AddSection()
For i As Integer = 0 To part.BodyItems.Count - 1
document.LastSection.Body.ChildEntities.Add(part.BodyItems(i))
Next
document.Save("Result.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Bookmarks.doc
x"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content
TextBodyPart part = bookmarkNavigator.GetBookmarkContent();
//Adds the retrieved content into another new section
document.AddSection();
for (int i = 0; i < part.BodyItems.Count; i++)
document.LastSection.Body.ChildEntities.Add(part.BodyItems[i]);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Closes the document
document.Close();

```

```
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Loads an existing Word document into DocIO instance
FileStream fileStreamPath = new FileStream("Bookmarks.docx", FileMode.Open,
    FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content
TextBodyPart part = bookmarkNavigator.GetBookmarkContent();
//Adds the retrieved content into another new section
document.AddSection();
for (int i = 0; i < part.BodyItems.Count; i++)
    document.LastSection.Body.ChildEntities.Add(part.BodyItems[i]);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
    WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Bookmarks.docx"),
        FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content
TextBodyPart part = bookmarkNavigator.GetBookmarkContent();
//Adds the retrieved content into another new section
document.AddSection();
for (int i = 0; i < part.BodyItems.Count; i++)
    document.LastSection.Body.ChildEntities.Add(part.BodyItems[i]);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
    "application/msword", stream);
```

```
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

The following code example shows how to retrieve the specified bookmark content by using `GetContent` method in a Word document.

C#

```
//Loads the template document with bookmark "Northwind" whose start and end
are preserved in different section.
WordDocument document = new WordDocument("Template.docx", FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content as WordDocumentPart
WordDocumentPart wordDocumentPart = bookmarkNavigator.GetContent();
//Saves the WordDocumentPart as separate Word document
WordDocument newDocument = wordDocumentPart.GetAsWordDocument();
//Close the WordDocumentPart instance
wordDocumentPart.Close();
//Close the template Word document
document.Close();
newDocument.Save("Result.docx", FormatType.Docx);
//Releases the resources hold by WordDocument instance
newDocument.Close();
```

VB.NET

```
'Loads the template document with bookmark "Northwind" whose start and end
are preserved in different section
Dim document As New WordDocument("Template.docx", FormatType.Docx)
'Creates the bookmark navigator instance to access the bookmark
Dim bookmarkNavigator As New BookmarksNavigator(document)
'Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind")
'Gets the bookmark content as WordDocumentPart
Dim wordDocumentPart As WordDocumentPart = bookmarkNavigator.GetContent()
'Saves the WordDocumentPart as separate Word document
Dim newDocument As WordDocument = wordDocumentPart.GetAsWordDocument()
'Close the WordDocumentPart instance
wordDocumentPart.Close()
'Close the template Word document
document.Close()
newDocument.Save("Result.docx", FormatType.Docx)
'Releases the resources hold by WordDocument instance
newDocument.Close()
```

UWP

```

//Loads the template document with bookmark "Northwind" whose start and end
are preserved in different section.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx
"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content as WordDocumentPart
WordDocumentPart wordDocumentPart = bookmarkNavigator.GetContent();
//Saves the WordDocumentPart as separate Word document
WordDocument newDocument = wordDocumentPart.GetAsWordDocument();
//Close the WordDocumentPart instance
wordDocumentPart.Close();
//Close the template Word document
document.Close();
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await newDocument.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Releases the resources hold by WordDocument instance
newDocument.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads the template document with bookmark "Northwind" whose start and end
are preserved in different section.
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content as WordDocumentPart
WordDocumentPart wordDocumentPart = bookmarkNavigator.GetContent();
//Saves the WordDocumentPart as separate Word document
WordDocument newDocument = wordDocumentPart.GetAsWordDocument();
//Close the WordDocumentPart instance
wordDocumentPart.Close();
//Closes the template document
document.Close();
MemoryStream stream = new MemoryStream();
newDocument.Save(stream, FormatType.Docx);
newDocument.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Loads the template document with bookmark "Northwind" whose start and end are preserved in different section.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Template.docx"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark "Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content as WordDocumentPart
WordDocumentPart wordDocumentPart = bookmarkNavigator.GetContent();
//Saves the WordDocumentPart as separate Word document
WordDocument newDocument = wordDocumentPart.GetAsWordDocument();
//Close the WordDocumentPart instance
wordDocumentPart.Close();
//Close the template document
document.Close();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
newDocument.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx", "application/msword", stream);
//Please download the helper files from the below link to save the stream as file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
//Releases the resources hold by WordDocument instance
newDocument.Close();

```

Inserting content into a bookmark

You can insert table, paragraph, simple text and paragraph item at the start or end location of the current bookmark by using bookmark navigator.

The following code example shows how to insert a simple text by using BookmarkNavigator.

C#

```

WordDocument document = new WordDocument("Bookmarks.docx", FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark "Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Inserts a new text before the bookmark end of the bookmark
bookmarkNavigator.InsertText(" Northwind Database is a set of tables containing data fitted into predefined categories.");
document.Save("Result.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

Dim document As New WordDocument("Bookmarks.docx", FormatType.Docx)
'Creates the bookmark navigator instance to access the bookmark
Dim bookmarkNavigator As New BookmarksNavigator(document)
'Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind")
'Inserts a new text before the bookmark end of the bookmark
bookmarkNavigator.InsertText(" Northwind Database is a set of tables
containing data fitted into predefined categories.")
document.Save("Result.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Bookmarks.doc
x"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Inserts a new text before the bookmark end of the bookmark
bookmarkNavigator.InsertText(" Northwind Database is a set of tables
containing data fitted into predefined categories.");
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads the template document
FileStream fileStreamPath = new FileStream(@"Bookmarks.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Inserts a new text before the bookmark end of the bookmark
bookmarkNavigator.InsertText(" Northwind Database is a set of tables
containing data fitted into predefined categories.");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);

```

```
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Bookmarks.docx"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Inserts a new text before the bookmark end of the bookmark
bookmarkNavigator.InsertText(" Northwind Database is a set of tables
containing data fitted into predefined categories.");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document.Close();
```

The following code example shows how to insert a paragraph item by using BookmarkNavigator.

C#

```
WordDocument document = new WordDocument("Bookmarks.docx", FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", false, true);
//Inserts a new picture after the bookmark end
WPicture picture =
bookmarkNavigator.InsertParagraphItem(ParagraphItemType.Picture) as
WPicture;
picture.LoadImage(Image.FromFile("Northwind.png"));
picture.WidthScale = 50;
picture.HeightScale = 50;
document.Save("Result.docx", FormatType.Docx);
document.Close();
```

VB.NET

```

Dim document As New WordDocument("Bookmarks.docx", FormatType.Docx)
'Creates the bookmark navigator instance to access the bookmark
Dim bookmarkNavigator As New BookmarksNavigator(document)
'Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", False, True)
'Inserts a new picture after the bookmark end
Dim picture As WPicture =
TryCast(bookmarkNavigator.InsertParagraphItem(ParagraphItemType.Picture),
WPicture)
picture.LoadImage(Image.FromFile("Northwind.png"))
picture.WidthScale = 50
picture.HeightScale = 50
document.Save("Result.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Bookmarks.doc
x"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", false, true);
//Inserts a new picture after the bookmark end
WPicture picture =
bookmarkNavigator.InsertParagraphItem(ParagraphItemType.Picture) as
WPicture;
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Northwind.png");
picture.LoadImage(imageStream);
picture.WidthScale = 50;
picture.HeightScale = 50;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads an existing Word document into DocIO instance
FileStream fileStreamPath = new FileStream("Bookmarks.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);

```



```
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", false, true);
//Inserts a new picture after the bookmark end
WPicture picture =
bookmarkNavigator.InsertParagraphItem(ParagraphItemType.Picture) as
WPicture;
FileStream imageStream = new FileStream("Northwind.png", FileMode.Open,
FileAccess.Read);
picture.LoadImage(imageStream);
picture.WidthScale = 50;
picture.HeightScale = 50;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Bookmarks.docx"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", false, true);
//Inserts a new picture after the bookmark end
WPicture picture =
bookmarkNavigator.InsertParagraphItem(ParagraphItemType.Picture) as
WPicture;
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("GettingStarted.Data.Northwind.png");
picture.LoadImage(imageStream);
picture.WidthScale = 50;
picture.HeightScale = 50;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
document.Close();
```

The following code example shows how to insert a paragraph by using BookmarkNavigator.

C#

```
WordDocument document = new WordDocument("Bookmarks.docx", FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarkNavigator bookmarkNavigator = new BookmarkNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", false, true);
//Inserts a new paragraph before the bookmark start
IWParagraph paragraph = new WParagraph(document);
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.");
bookmarkNavigator.InsertParagraph(paragraph);
document.Save("Result.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
Dim document As New WordDocument("Bookmarks.docx", FormatType.Docx)
'Creates the bookmark navigator instance to access the bookmark
Dim bookmarkNavigator As New BookmarkNavigator(document)
'Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", False, True)
'Inserts a new paragraph before the bookmark start
Dim paragraph As IWParagraph = New WParagraph(document)
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.")
bookmarkNavigator.InsertParagraph(paragraph)
document.Save("Result.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Bookmarks.doc
x"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarkNavigator bookmarkNavigator = new BookmarkNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", false, true);
//Inserts a new paragraph before the bookmark start
IWParagraph paragraph = new WParagraph(document);
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.");
bookmarkNavigator.InsertParagraph(paragraph);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
```

```
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Loads an existing Word document into DocIO instance
FileStream filePath = new FileStream("Bookmarks.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(filePath, FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", false, true);
//Inserts a new paragraph before the bookmark start
IWParagraph paragraph = new WParagraph(document);
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.");
bookmarkNavigator.InsertParagraph(paragraph);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Bookmar
ks.docx"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", false, true);
//Inserts a new paragraph before the bookmark start
IWParagraph paragraph = new WParagraph(document);
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.");
bookmarkNavigator.InsertParagraph(paragraph);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
```

```
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
document.Close();
```

The following code example shows how to insert a table by using BookmarkNavigator.

C#

```
WordDocument document = new WordDocument("Bookmarks.docx", FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarkNavigator bookmarkNavigator = new BookmarkNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", false, false);
//Inserts a new paragraph before the bookmark end
IWParagraph paragraph = new WParagraph(document);
paragraph.AppendText("Northwind Database Contains the following tables:");
bookmarkNavigator.InsertParagraph(paragraph);
//Inserts a new table before the bookmark end
WTable table = new WTable(document);
table.ResetCells(3, 2);
table[0, 0].AddParagraph().AppendText("Suppliers");
table[0, 1].AddParagraph().AppendText("2");
table[1, 0].AddParagraph().AppendText("Customers");
table[1, 1].AddParagraph().AppendText("1");
table[2, 0].AddParagraph().AppendText("Employees");
table[2, 1].AddParagraph().AppendText("3");
bookmarkNavigator.InsertTable(table);
document.Save("Result.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
Dim document As New WordDocument("Bookmarks.docx", FormatType.Docx)
'Creates the bookmark navigator instance to access the bookmark
Dim bookmarkNavigator As New BookmarkNavigator(document)
'Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", False, False)
'Inserts a new paragraph before the bookmark end
Dim paragraph As IWParagraph = New WParagraph(document)
paragraph.AppendText("Northwind Database Contains the following tables:")
bookmarkNavigator.InsertParagraph(paragraph)
'Inserts a new table before the bookmark end
Dim table As New WTable(document)
table.ResetCells(3, 2)
table(0, 0).AddParagraph().AppendText("Suppliers")
table(0, 1).AddParagraph().AppendText("2")
table(1, 0).AddParagraph().AppendText("Customers")
table(1, 1).AddParagraph().AppendText("1")
table(2, 0).AddParagraph().AppendText("Employees")
table(2, 1).AddParagraph().AppendText("3")
bookmarkNavigator.InsertTable(table)
document.Save("Result.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Bookmarks.doc
x"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", false, false);
//Inserts a new paragraph before the bookmark end
IWParagraph paragraph = new WParagraph(document);
paragraph.AppendText("Northwind Database Contains the following tables:");
bookmarkNavigator.InsertParagraph(paragraph);
//Inserts a new table before the bookmark end
WTable table = new WTable(document);
table.ResetCells(3, 2);
table[0, 0].AddParagraph().AppendText("Suppliers");
table[0, 1].AddParagraph().AppendText("2");
table[1, 0].AddParagraph().AppendText("Customers");
table[1, 1].AddParagraph().AppendText("1");
table[2, 0].AddParagraph().AppendText("Employees");
table[2, 1].AddParagraph().AppendText("3");
bookmarkNavigator.InsertTable(table);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Loads an existing Word document into DocIO instance
FileStream fileStreamPath = new FileStream("Bookmarks.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", false, false);
//Inserts a new paragraph before the bookmark end
IWParagraph paragraph = new WParagraph(document);
paragraph.AppendText("Northwind Database Contains the following tables:");
bookmarkNavigator.InsertParagraph(paragraph);
//Inserts a new table before the bookmark end
WTable table = new WTable(document);
table.ResetCells(3, 2);
table[0, 0].AddParagraph().AppendText("Suppliers");
table[0, 1].AddParagraph().AppendText("2");
```

```

table[1, 0].AddParagraph().AppendText("Customers");
table[1, 1].AddParagraph().AppendText("1");
table[2, 0].AddParagraph().AppendText("Employees");
table[2, 1].AddParagraph().AppendText("3");
bookmarkNavigator.InsertTable(table);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Bookmarks.docx"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind", false, false);
//Inserts a new paragraph before the bookmark end
IWParagraph paragraph = new WParagraph(document);
paragraph.AppendText("Northwind Database Contains the following tables:");
bookmarkNavigator.InsertParagraph(paragraph);
//Inserts a new table before the bookmark end
WTable table = new WTable(document);
table.ResetCells(3, 2);
table[0, 0].AddParagraph().AppendText("Suppliers");
table[0, 1].AddParagraph().AppendText("2");
table[1, 0].AddParagraph().AppendText("Customers");
table[1, 1].AddParagraph().AppendText("1");
table[2, 0].AddParagraph().AppendText("Employees");
table[2, 1].AddParagraph().AppendText("3");
bookmarkNavigator.InsertTable(table);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document.Close();

```

The following code example shows how to insert a TextBodyPart by using BookmarkNavigator.

C#

```

WordDocument document = new WordDocument("Bookmarks.docx", FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content
TextBodyPart textBodyPart = bookmarkNavigator.GetBookmarkContent();
document.AddSection();
IWParagraph paragraph = document.LastSection.AddParagraph();
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.");
//Adds the new bookmark into Word document
paragraph.AppendBookmarkStart("bookmark_empty");
paragraph.AppendBookmarkEnd("bookmark_empty");
//Moves the virtual cursor to the location after the start of the bookmark
"bookmark_empty"
bookmarkNavigator.MoveToBookmark("bookmark_empty", true, true);
//Inserts the text body part after the bookmark start
bookmarkNavigator.InsertTextBodyPart(textBodyPart);
document.Save("Result.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

Dim document As New WordDocument("Bookmarks.docx", FormatType.Docx)
'Creates the bookmark navigator instance to access the bookmark
Dim bookmarkNavigator As New BookmarksNavigator(document)
'Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind")
'Gets the bookmark content
Dim textBodyPart As TextBodyPart = bookmarkNavigator.GetBookmarkContent()
document.AddSection()
Dim paragraph As IWParagraph = document.LastSection.AddParagraph()
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.")
'Adds the new bookmark into Word document
paragraph.AppendBookmarkStart("bookmark_empty")
paragraph.AppendBookmarkEnd("bookmark_empty")
'Moves the virtual cursor to the location after the start of the bookmark
"bookmark_empty"
bookmarkNavigator.MoveToBookmark("bookmark_empty", True, True)
'Inserts the text body part after the bookmark start
bookmarkNavigator.InsertTextBodyPart(textBodyPart)
document.Save("Result.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Bookmarks.doc
x"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark

```

```

BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content
TextBodyPart textBodyPart = bookmarkNavigator.GetBookmarkContent();
document.AddSection();
IWParagraph paragraph = document.LastSection.AddParagraph();
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.");
//Adds the new bookmark into Word document
paragraph.AppendBookmarkStart("bookmark_empty");
paragraph.AppendBookmarkEnd("bookmark_empty");
//Moves the virtual cursor to the location after the start of the bookmark
"bookmark_empty"
bookmarkNavigator.MoveToBookmark("bookmark_empty", true, true);
//Inserts the text body part after the bookmark start
bookmarkNavigator.InsertTextBodyPart(textBodyPart);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads an existing Word document into DocIO instance
FileStream fileStreamPath = new FileStream("Bookmarks.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content
TextBodyPart textBodyPart = bookmarkNavigator.GetBookmarkContent();
document.AddSection();
IWParagraph paragraph = document.LastSection.AddParagraph();
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.");
//Adds the new bookmark into Word document
paragraph.AppendBookmarkStart("bookmark_empty");
paragraph.AppendBookmarkEnd("bookmark_empty");
//Moves the virtual cursor to the location after the start of the bookmark
"bookmark_empty"
bookmarkNavigator.MoveToBookmark("bookmark_empty", true, true);
//Inserts the text body part after the bookmark start
bookmarkNavigator.InsertTextBodyPart(textBodyPart);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);

```



```
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Bookmarks.docx"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content
TextBodyPart textBodyPart = bookmarkNavigator.GetBookmarkContent();
document.AddSection();
IWParagraph paragraph = document.LastSection.AddParagraph();
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.");
//Adds the new bookmark into Word document
paragraph.AppendBookmarkStart("bookmark_empty");
paragraph.AppendBookmarkEnd("bookmark_empty");
//Moves the virtual cursor to the location after the start of the bookmark
"bookmark_empty"
bookmarkNavigator.MoveToBookmark("bookmark_empty", true, true);
//Inserts the text body part after the bookmark start
bookmarkNavigator.InsertTextBodyPart(textBodyPart);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Deleting content from a bookmark

You can delete the contents between bookmark start and end of the specified bookmark in a Word document.

The following code example shows how to remove the contents of a specified bookmark from Word document.

C#

```
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument("Bookmarks.docx", FormatType.Docx);
```

```
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind "
bookmarkNavigator.MoveToBookmark("Northwind");
//Deletes bookmark content without deleting the format in the target
document.
bookmarkNavigator.DeleteBookmarkContent(false);
document.Save("Result.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Loads an existing Word document into DocIO instance
Dim document As New WordDocument("Bookmarks.docx", FormatType.Docx)
'Creates the bookmark navigator instance to access the bookmark
Dim bookmarkNavigator As New BookmarksNavigator(document)
'Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind")
'Deletes bookmark content without deleting the format in the target
document.
bookmarkNavigator.DeleteBookmarkContent(False)
document.Save("Result.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Bookmarks.doc
x"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind "
bookmarkNavigator.MoveToBookmark("Northwind");
//Deletes bookmark content without deleting the format in the target
document.
bookmarkNavigator.DeleteBookmarkContent(false);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Loads an existing Word document into DocIO instance
```

```

FileStream filePath = new FileStream("Bookmarks.docx", FileMode.Open,
    FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(filePath, FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind "
bookmarkNavigator.MoveToBookmark("Northwind");
//Deletes bookmark content without deleting the format in the target
document.
bookmarkNavigator.DeleteBookmarkContent(false);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Bookmar
ks.docx"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind "
bookmarkNavigator.MoveToBookmark("Northwind");
//Deletes bookmark content without deleting the format in the target
document.
bookmarkNavigator.DeleteBookmarkContent(false);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Replacing content in a bookmark

You can replace the contents of an existing bookmark with simple text, TextBodyPart, WordDocumentPart.

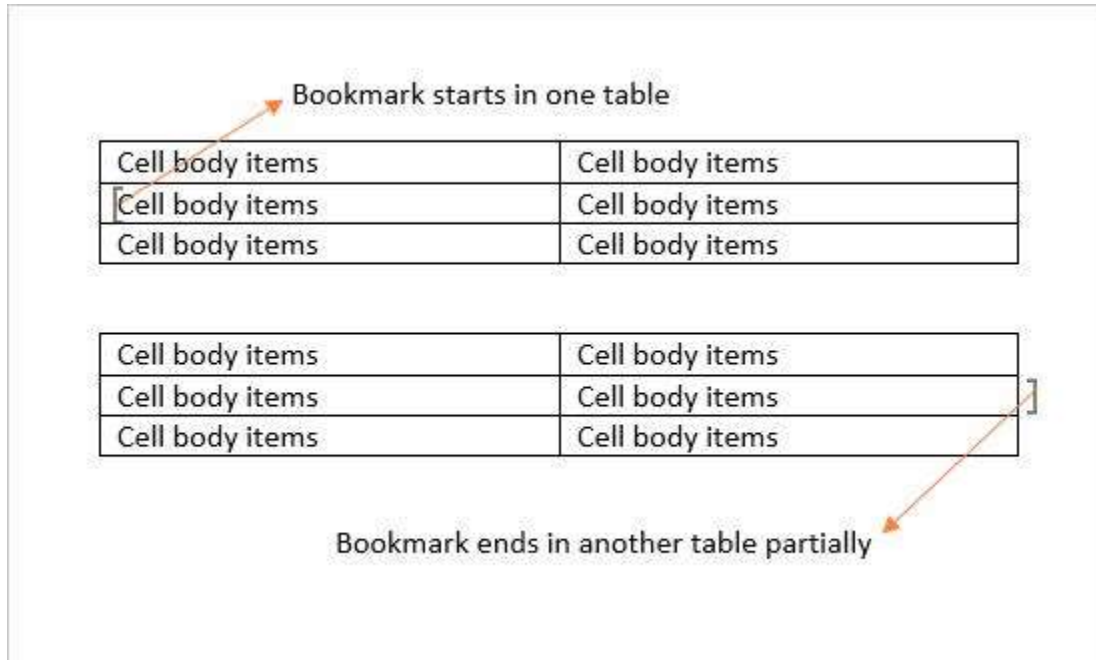
Note:

You cannot replace the multi section contents into a bookmark within table in Word documents. Use "for loop" instead of "foreach loop" to iterate through document elements when replacing the

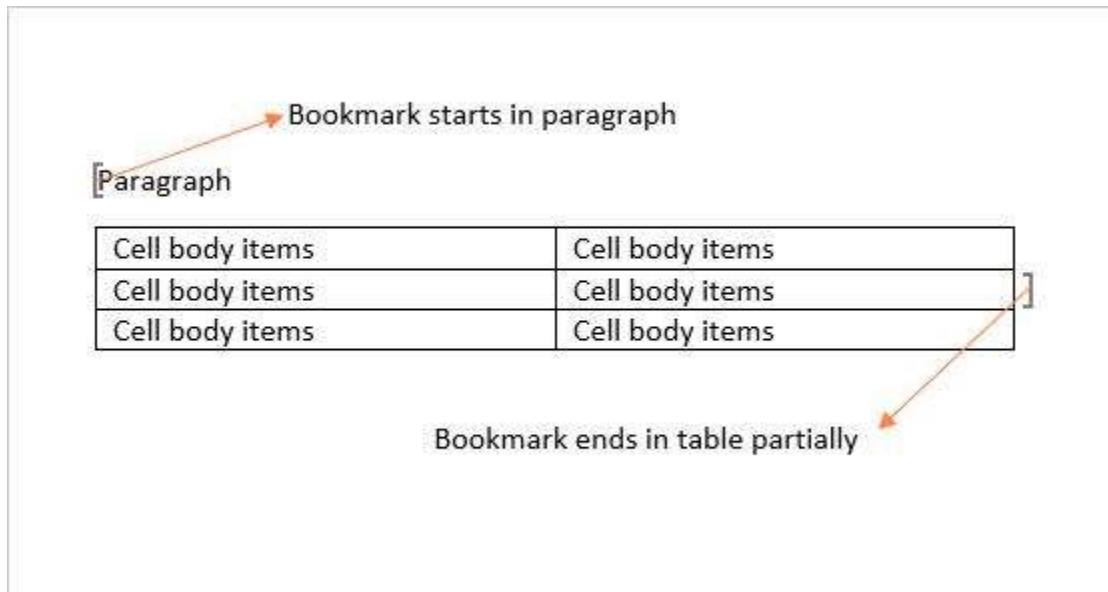
bookmark contents to avoid “collection modified exception”, as there is a chance for modification in the document elements on replacing the bookmark contents.

As per Microsoft Word behavior, you cannot replace the bookmark contents when the bookmark start and end is not in a same table as following cases:

Case 1



Case 2



The following code example shows how to replace a specified bookmark content by using `ReplaceBookmarkContent` method in Word document.

C#

```

WordDocument document = new WordDocument("Bookmarks.docx", FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content
TextBodyPart textBodyPart = bookmarkNavigator.GetBookmarkContent();
document.AddSection();
IWParagraph paragraph = document.LastSection.AddParagraph();
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.");
//Adds the new bookmark into Word document
paragraph.AppendBookmarkStart("bookmark_empty");
paragraph.AppendBookmarkEnd("bookmark_empty");
//Moves the virtual cursor to the location before the end of the bookmark
"bookmark_empty"
bookmarkNavigator.MoveToBookmark("bookmark_empty");
//Replaces the bookmark content with text body part
bookmarkNavigator.ReplaceBookmarkContent(textBodyPart);
document.Save("Result.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

Dim document As New WordDocument("Bookmarks.docx", FormatType.Docx)
'Creates the bookmark navigator instance to access the bookmark
Dim bookmarkNavigator As New BookmarksNavigator(document)
'Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind")
'Gets the bookmark content
Dim textBodyPart As TextBodyPart = bookmarkNavigator.GetBookmarkContent()
document.AddSection()
Dim paragraph As IWParagraph = document.LastSection.AddParagraph()
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.")
'Adds the new bookmark into Word document
paragraph.AppendBookmarkStart("bookmark_empty")
paragraph.AppendBookmarkEnd("bookmark_empty")
'Moves the virtual cursor to the location before the end of the bookmark
"bookmark_empty"
bookmarkNavigator.MoveToBookmark("bookmark_empty")
'Replaces the bookmark content with text body part
bookmarkNavigator.ReplaceBookmarkContent(textBodyPart)
document.Save("Result.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Bookmarks.doc
x"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark

```

```

BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content
TextBodyPart textBodyPart = bookmarkNavigator.GetBookmarkContent();
document.AddSection();
IWParagraph paragraph = document.LastSection.AddParagraph();
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.");
//Adds the new bookmark into Word document
paragraph.AppendBookmarkStart("bookmark_empty");
paragraph.AppendBookmarkEnd("bookmark_empty");
//Moves the virtual cursor to the location before the end of the bookmark
"bookmark_empty"
bookmarkNavigator.MoveToBookmark("bookmark_empty");
//Replaces the bookmark content with text body part
bookmarkNavigator.ReplaceBookmarkContent(textBodyPart);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads an existing Word document into DocIO instance
FileStream fileStreamPath = new FileStream("Bookmarks.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content
TextBodyPart textBodyPart = bookmarkNavigator.GetBookmarkContent();
document.AddSection();
IWParagraph paragraph = document.LastSection.AddParagraph();
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.");
//Adds the new bookmark into Word document
paragraph.AppendBookmarkStart("bookmark_empty");
paragraph.AppendBookmarkEnd("bookmark_empty");
//Moves the virtual cursor to the location before the end of the bookmark
"bookmark_empty"
bookmarkNavigator.MoveToBookmark("bookmark_empty");
//Replaces the bookmark content with text body part
bookmarkNavigator.ReplaceBookmarkContent(textBodyPart);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);

```

```
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Bookmarks.docx"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content
TextBodyPart textBodyPart = bookmarkNavigator.GetBookmarkContent();
document.AddSection();
IWParagraph paragraph = document.LastSection.AddParagraph();
paragraph.AppendText("Northwind Database is a set of tables containing data
fitted into predefined categories.");
//Adds the new bookmark into Word document
paragraph.AppendBookmarkStart("bookmark_empty");
paragraph.AppendBookmarkEnd("bookmark_empty");
//Moves the virtual cursor to the location before the end of the bookmark
"bookmark_empty"
bookmarkNavigator.MoveToBookmark("bookmark_empty");
//Replaces the bookmark content with text body part
bookmarkNavigator.ReplaceBookmarkContent(textBodyPart);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

The following code example shows how to replace a specified bookmark content by using `ReplaceContent` method in Word document.

C#

```
//Loads the template document with bookmark "Northwind" whose start and end
are preserved in different section
WordDocument templateDocument = new WordDocument("Template.docx",
FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new
BookmarksNavigator(templateDocument);
```

```

//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content as WordDocumentPart
WordDocumentPart wordDocumentPart = bookmarkNavigator.GetContent();
//Loads the Word document with bookmark NorthwindDB
WordDocument document = new WordDocument("Bookmarks.docx", FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"NorthwindDB"
bookmarkNavigator.MoveToBookmark("NorthwindDB");
//Replaces the bookmark content with word body part
bookmarkNavigator.ReplaceContent(wordDocumentPart);
//Close the WordDocumentPart instance
wordDocumentPart.Close();
//Closes the template document
templateDocument.Close();
document.Save("Result.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Loads the template document with bookmark "Northwind" whose start and end
are preserved in different section
Dim templateDocument As New WordDocument("Template.docx", FormatType.Docx)
'Creates the bookmark navigator instance to access the bookmark
Dim bookmarkNavigator As New BookmarksNavigator(templateDocument)
'Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind")
'Gets the bookmark content as WordDocumentPart
Dim wordDocumentPart As WordDocumentPart = bookmarkNavigator.GetContent()
'Loads the Word document with bookmark NorthwindDB
Dim document As New WordDocument("Bookmarks.docx", FormatType.Docx)
'Creates the bookmark navigator instance to access the bookmark
bookmarkNavigator = New BookmarksNavigator(document)
'Moves the virtual cursor to the location before the end of the bookmark
"NorthwindDB"
bookmarkNavigator.MoveToBookmark("NorthwindDB")
'Replaces the bookmark content with word body part
bookmarkNavigator.ReplaceContent(wordDocumentPart)
'Close the WordDocumentPart instance
wordDocumentPart.Close()
'Closes the template document
templateDocument.Close()
document.Save("Result.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Loads the template document with bookmark "Northwind" whose start and end
are preserved in different section
Assembly assembly = typeof(App).GetTypeInfo().Assembly;

```



```

WordDocument templateDocument = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx
"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new
BookmarksNavigator(templateDocument);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content as WordDocumentPart
WordDocumentPart wordDocumentPart = bookmarkNavigator.GetContent();
//Loads the Word document with bookmark NorthwindDB
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Bookmarks.doc
x"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"NorthwindDB"
bookmarkNavigator.MoveToBookmark("NorthwindDB");
//Replaces the bookmark content with word body part
bookmarkNavigator.ReplaceContent(wordDocumentPart);
//Close the WordDocumentPart instance
wordDocumentPart.Close();
//Closes the template document
templateDocument.Close();
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads the template document with bookmark "Northwind" whose start and end
are preserved in different section
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument templateDocument = new WordDocument(fileStreamPath,
FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new
BookmarksNavigator(templateDocument);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content as WordDocumentPart
WordDocumentPart wordDocumentPart = bookmarkNavigator.GetContent();
//Loads the Word document with bookmark NorthwindDB
FileStream fileStreamPath = new FileStream("Bookmarks.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);

```

```
//Creates the bookmark navigator instance to access the bookmark
bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"NorthwindDB"
bookmarkNavigator.MoveToBookmark("NorthwindDB");
//Replaces the bookmark content with word body part
bookmarkNavigator.ReplaceContent(wordDocumentPart);
//Close the WordDocumentPart instance
wordDocumentPart.Close();
//Closes the template document
templateDocument.Close();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Loads the template document with bookmark "Northwind" whose start and end
are preserved in different section
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument templateDocument = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Templat
e.docx"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
BookmarksNavigator bookmarkNavigator = new
BookmarksNavigator(templateDocument);
//Moves the virtual cursor to the location before the end of the bookmark
"Northwind"
bookmarkNavigator.MoveToBookmark("Northwind");
//Gets the bookmark content as WordDocumentPart
WordDocumentPart wordDocumentPart = bookmarkNavigator.GetContent();
//Loads the Word document with bookmark NorthwindDB
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Bookmar
ks.docx"), FormatType.Docx);
//Creates the bookmark navigator instance to access the bookmark
bookmarkNavigator = new BookmarksNavigator(document);
//Moves the virtual cursor to the location before the end of the bookmark
"NorthwindDB"
bookmarkNavigator.MoveToBookmark("NorthwindDB");
//Replaces the bookmark content with word body part
bookmarkNavigator.ReplaceContent(wordDocumentPart);
//Close the WordDocumentPart instance
wordDocumentPart.Close();
//Closes the template document
templateDocument.Close();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
```

```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Working with document Fields

Fields in a Word document are placeholders for data that might change on field update. Fields are represented by the `WField` and `WFieldMark` instances in DocIO. A field in a Word document contains field codes, field separator, field result, and field end.

To learn various types of Microsoft Word supported fields and their syntax, refer to the [MSDN article](#)

From v16.1.0.24, the entire field code is included in Document Object Model(DOM). Hence, adding a field will automatically include the following elements in DOM:

1. `WField`: Represents the starting of a Field.
2. `ParagraphItem`: Represents the Field code.
3. `WFieldMark`: Represents the Field separator.
4. `ParagraphItem`: Represents the Field result.
5. `WFieldMark`: Represents the end of a Field.

Find more information about migration changes [here](#).

Adding fields

You can add a field to a Word document by using the `AppendField` method of `WParagraph` class.

The following code example explains how to add a field to the Word document.

C#

```
//Creates an instance of WordDocument class (Empty Word Document)
WordDocument document = new WordDocument();
//Adds a new section to the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph to Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Today's Date: ");
//Adds the new Date field to Word document with field name and its type
WField field = paragraph.AppendField("Date", FieldType.FieldDate) as WField;
//Field code used to describe how to display the date
field.FieldCode = @"DATE \@\" + "\"MMMM d, yyyy\"";
//Saves the document in the given name and format
document.Save("Sample.docx", FormatType.Docx);
//Releases the resources occupied by WordDocument instance
document.Close();
```

VB.NET

```
'Creates an instance of WordDocument class (Empty Word Document)
Dim document As New WordDocument()
```

```

'Adds a new section to the Word Document
Dim section As IWSection = document.AddSection()
'Adds a new paragraph to Word document and appends text into paragraph
Dim paragraph As IWParagraph = section.AddParagraph()
paragraph.AppendText("Today's Date: ")
'Adds the new Date field to Word document with field name and its type
Dim field As WField = TryCast(paragraph.AppendField("Date",
FieldType.FieldDate), WField)
'Field code used to describe how to display the date
field.FieldCode = "DATE \@ + \"\"MMMM d, yyyy\"\"
'Saves the document in the given name and format
document.Save("Sample.docx", FormatType.Docx)
'Releases the resources occupied by WordDocument instance
document.Close()

```

UWP

```

//Creates an instance of WordDocument class (Empty Word Document)
WordDocument document = new WordDocument();
//Adds a new section to the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph to Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Today's Date: ");
//Adds the new Date field to Word document with field name and its type
WField field = paragraph.AppendField("Date", FieldType.FieldDate) as WField;
//Field code used to describe how to display the date
field.FieldCode = @"DATE \@ + \"\"MMMM d, yyyy\"\";
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Refer to the following link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class (Empty Word Document)
WordDocument document = new WordDocument();
//Adds a new section to the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph to Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Today's Date: ");
//Adds the new Date field to Word document with field name and its type
WField field = paragraph.AppendField("Date", FieldType.FieldDate) as WField;
//Field code used to describe how to display the date
field.FieldCode = @"DATE \@ + \"\"MMMM d, yyyy\"\";
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document

```

```
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Creates an instance of WordDocument class (Empty Word Document)
WordDocument document = new WordDocument();
//Adds a new section to the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph to Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Today's Date: ");
//Adds the new Date field to Word document with field name and its type
WField field = paragraph.AppendField("Date", FieldType.FieldDate) as WField;
//Field code used to describe how to display the date
field.FieldCode = @"DATE \@\" + "\"MMMM d, yyyy\"";
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Formatting fields

You can format the field instances added to the Word document by iterating the items from field start to end.

The following code example explains how to format the field in Word document.

C#

```
//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Adds the new Page field to Word document with field name and its type
IWField field = document.LastParagraph.AppendField("Page",
FieldType.FieldPage);
IEntity entity = field;
//Iterates to sibling items until Field End
while (entity.NextSibling != null)
{
    if (entity is WTextRange)
    //Sets character format for text ranges
    (entity as WTextRange).CharacterFormat.FontSize = 6;
    else if ((entity is WFieldMark) && (entity as WFieldMark).Type ==
FieldMarkType.FieldEnd)
    break;
```

```
//Gets next sibling item
entity = entity.NextSibling;
}
//Saves and closes the WordDocument instance
document.Save("Template.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates an instance of a WordDocument
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Adds the new Page field in Word document with field name and its type
Dim field As IField = document.LastParagraph.AppendField("Page",
FieldType.FieldPage)
Dim entity As IEntity = field
'Iterates to sibling items until Field End
While (Not (entity.NextSibling) Is Nothing)
'Sets character format for text ranges
If (TypeOf entity Is WTextRange) Then
CType(entity, WTextRange).CharacterFormat.FontSize = 6
ElseIf ((TypeOf entity Is WFieldMark) AndAlso (CType(entity,
WFieldMark).Type = FieldMarkType.FieldEnd)) Then
Exit While
End If
'Gets next sibling item
entity = entity.NextSibling
End While
'Saves and closes the WordDocument instance
document.Save("Template.docx", FormatType.Docx)
document.Close
```

UWP

```
//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Adds the new Page field to Word document with field name and its type
IField field = document.LastParagraph.AppendField("Page",
FieldType.FieldPage);
IEntity entity = field;
//Iterates to sibling items until Field End
while (entity.NextSibling != null)
{
if (entity is WTextRange)
//Sets character format for text ranges
(entity as WTextRange).CharacterFormat.FontSize = 6;
else if ((entity is WFieldMark) && (entity as WFieldMark).Type ==
FieldMarkType.FieldEnd)
break;
//Gets next sibling item.
entity = entity.NextSibling;
}
//Saves the Word file to MemoryStream
```

```

MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Template.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Adds the new Page field in Word document with field name and its type
IWField field = document.LastParagraph.AppendField("Page",
FieldType.FieldPage);
IEntity entity = field;
//Iterates to sibling items until Field End
while (entity.NextSibling != null)
{
    if (entity is WTextRange)
    //Sets character format for text ranges
    (entity as WTextRange).CharacterFormat.FontSize = 6;
    else if ((entity is WFieldMark) && (entity as WFieldMark).Type ==
FieldMarkType.FieldEnd)
    break;
    //Gets next sibling item.
    entity = entity.NextSibling;
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Template.docx");

```

XAMARIN

```

//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Adds the new Page field to Word document with field name and its type
IWField field = document.LastParagraph.AppendField("Page",
FieldType.FieldPage);
IEntity entity = field;
//Iterates to sibling items until Field End
while (entity.NextSibling != null)
{
    if (entity is WTextRange)
    //Sets character format for text ranges

```

```

(entity as WTextRange).CharacterFormat.FontSize = 6;
else if ((entity is WFieldMark) && (entity as WFieldMark).Type ==
FieldMarkType.FieldEnd)
break;
//Gets next sibling item
entity = entity.NextSibling;
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Template.docx",
"application/msword", stream);
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Updating fields

Field updating engine calculates the resultant value based on the field code information and updates the field result with a new value. You can update the following fields by using DocIO:

- = (formula field)
- DATE
- TIME
- DOCVARIABLE
- DOCPROPERTY
- COMPARE
- IF
- NEXTIF
- MERGEREC
- MERGESEQ
- SECTION
- NUMPAGES
- TITLE
- Cross-Reference

The following are the known limitations:

- Updating of NUMPAGES field and Cross Reference field with Page number and Paragraph number options are not supported in Silverlight, WinRT, Universal, Windows Phone, and Xamarin applications.
- Currently group shapes, drawing canvas, and table auto resizing are not supported in Word to PDF layouting, and this may lead to update incorrect page number and total number of pages.

The following code example explains how to update the fields present in Word document.

C#


```
//Loads an existing Word document into DocIO instance
WordDocument document = new WordDocument("Input.docx", FormatType.Docx);
//Updates the fields present in a document
document.UpdateDocumentFields();
document.Save("Result.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Loads an existing Word document into DocIO instance
Dim document As New WordDocument("Input.docx", FormatType.Docx)
'Updates the fields present in a document
document.UpdateDocumentFields()
document.Save("Result.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Input.docx"),
FormatType.Docx);
//Updates the fields present in a document
document.UpdateDocumentFields();
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Loads an existing Word document into DocIO instance
FileStream fileStreamPath = new FileStream("Input.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Updates the fields present in a document
document.UpdateDocumentFields();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Loads an existing Word document into DocIO instance
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Input.docx"), FormatType.Docx);
//Updates the fields present in a document
document.UpdateDocumentFields();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

IF field

IF field compares two values and updates the field result with true text, when comparison succeeds otherwise false text.

To learn more about IF field and its syntax in Microsoft Word, refer to the [MSDN article](#)

The following code example explains how to add an If field to a Word document.

C#

```
//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("If field which uses string of characters in
expression");
paragraph = section.AddParagraph();
//Creates the new instance of IF field
WIfField field = paragraph.AppendField("If", FieldType.FieldIf) as WIfField;
//Specifies the expression, true and false statement in field code
field.FieldCode = "IF \"True\" = \"True\" \"The given statement is Correct\"
\"The given statement is Wrong\"";
paragraph = section.AddParagraph();
paragraph.AppendText("If field which uses numbers in expression");
paragraph = section.AddParagraph();
//Creates the new instance of IF field
field = paragraph.AppendField("If", FieldType.FieldIf) as WIfField;
//Specifies the expression, true and false statement in field code
field.FieldCode = "IF 100 >= 1000 \"The given statement is Correct\" \"The
given statement is Wrong\"";
//Updates the document fields
document.UpdateDocumentFields();
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```

'Creates an instance of a WordDocument
Dim document As New WordDocument()
Dim section As IWSection = document.AddSection()
Dim paragraph As IWParagraph = section.AddParagraph()
paragraph.AppendText("If field which uses string of characters in
expression")
paragraph = section.AddParagraph()
'Creates the new instance of IF field
Dim field As WIfField = TryCast(paragraph.AppendField("If",
FieldType.FieldIf), WIfField)
'Specifies the expression, true and false statement in field code
field.FieldCode = "IF ""True"" = ""True"" ""The given statement is Correct""
""The given statement is Wrong"""
paragraph = section.AddParagraph()
paragraph.AppendText("If field which uses numbers in expression")
paragraph = section.AddParagraph()
'Creates the new instance of IF field
field = TryCast(paragraph.AppendField("If", FieldType.FieldIf), WIfField)
'Specify the expression, true and false statement in field code
field.FieldCode = "IF 100 >= 1000 ""The given statement is Correct"" ""The
given statement is Wrong"""
'Updates the document fields
document.UpdateDocumentFields()
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("If field which uses string of characters in
expression");
paragraph = section.AddParagraph();
//Creates the new instance of IF field
WIfField field = paragraph.AppendField("If", FieldType.FieldIf) as WIfField;
//Specifies the expression, true and false statement in field code
field.FieldCode = "IF \"True\" = \"True\" \"The given statement is Correct\"
\"The given statement is Wrong\"";
paragraph = section.AddParagraph();
paragraph.AppendText("If field which uses numbers in expression");
paragraph = section.AddParagraph();
//Creates the new instance of IF field
field = paragraph.AppendField("If", FieldType.FieldIf) as WIfField;
//Specifies the expression, true and false statement in field code
field.FieldCode = "IF 100 >= 1000 \"The given statement is Correct\" \"The
given statement is Wrong\"";
//Updates the document fields
document.UpdateDocumentFields();
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine

```

```
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("If field which uses string of characters in expression");
paragraph = section.AddParagraph();
//Creates the new instance of IF field
WifField field = paragraph.AppendField("If", FieldType.FieldIf) as WifField;
//Specifies the expression, true and false statement in field code
field.FieldCode = "IF \"True\" = \"True\" \"The given statement is Correct\" \"The given statement is Wrong\"";
paragraph = section.AddParagraph();
paragraph.AppendText("If field which uses numbers in expression");
paragraph = section.AddParagraph();
//Creates the new instance of IF field
field = paragraph.AppendField("If", FieldType.FieldIf) as WifField;
//Specifies the expression, true and false statement in field code
field.FieldCode = "IF 100 >= 1000 \"The given statement is Correct\" \"The given statement is Wrong\"";
//Updates the document fields
document.UpdateDocumentFields();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("If field which uses string of characters in expression");
paragraph = section.AddParagraph();
//Creates the new instance of IF field
WifField field = paragraph.AppendField("If", FieldType.FieldIf) as WifField;
//Specifies the expression, true and false statement in field code
```

```

field.FieldCode = "IF \"True\" = \"True\" \"The given statement is Correct\"
\"The given statement is Wrong\"";
paragraph = section.AddParagraph();
paragraph.AppendText("If field which uses numbers in expression");
paragraph = section.AddParagraph();
//Creates the new instance of IF field
field = paragraph.AppendField("If", FieldType.FieldIf) as WIfField;
//Specifies the expression, true and false statement in field code
field.FieldCode = "IF 100 >= 1000 \"The given statement is Correct\" \"The
given statement is Wrong\"";
//Updates the document fields
document.UpdateDocumentFields();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Document variables

The DocVariable field displays the value of a specified document variable in the Word document. The document variables can be added or modified using the **Variables** property of **WordDocument** class.

The following code example explains how to add a DocVariable field to a Word document.

C#

```

//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("First Name of the customer: ");
//Adds the DocVariable field with Variable name and its type
paragraph.AppendField("FirstName", FieldType.FieldDocVariable);
paragraph = section.AddParagraph();
paragraph.AppendText("Last Name of the customer: ");
//Adds the DocVariable field with Variable name and its type
paragraph.AppendField("LastName", FieldType.FieldDocVariable);
//Adds the value for variable in WordDocument.Variable collection
document.Variables.Add("FirstName", "Jeff");
document.Variables.Add("LastName", "Smith");
//Updates the document fields
document.UpdateDocumentFields();
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates an instance of a WordDocument
Dim document As New WordDocument()

```

```

Dim section As IWSection = document.AddSection()
Dim paragraph As IWParagraph = section.AddParagraph()
paragraph.AppendText("First Name of the customer: ")
'Adds the DocVariable field with Variable name and its type
paragraph.AppendField("FirstName", FieldType.FieldDocVariable)
paragraph = section.AddParagraph()
paragraph.AppendText("Last Name of the customer: ")
'Adds the DocVariable field with Variable name and its type
paragraph.AppendField("LastName", FieldType.FieldDocVariable)
'Adds the value for variable in WordDocument.Variable collection
document.Variables.Add("FirstName", "Jeff")
document.Variables.Add("LastName", "Smith")
'Updates the document fields
document.UpdateDocumentFields()
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("First Name of the customer: ");
//Adds the DocVariable field with Variable name and its type
paragraph.AppendField("FirstName", FieldType.FieldDocVariable);
paragraph = section.AddParagraph();
paragraph.AppendText("Last Name of the customer: ");
//Adds the DocVariable field with Variable name and its type
paragraph.AppendField("LastName", FieldType.FieldDocVariable);
//Adds the value for variable in WordDocument.Variable collection
document.Variables.Add("FirstName", "Jeff");
document.Variables.Add("LastName", "Smith");
//Updates the document fields
document.UpdateDocumentFields();
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();

```

```

paragraph.AppendText("First Name of the customer: ");
//Adds the DocVariable field with Variable name and its type
paragraph.AppendField("FirstName", FieldType.FieldDocVariable);
paragraph = section.AddParagraph();
paragraph.AppendText("Last Name of the customer: ");
//Adds the DocVariable field with Variable name and its type
paragraph.AppendField("LastName", FieldType.FieldDocVariable);
//Adds the value for variable in WordDocument.Variable collection
document.Variables.Add("FirstName", "Jeff");
document.Variables.Add("LastName", "Smith");
//Updates the document fields
document.UpdateDocumentFields();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("First Name of the customer: ");
//Adds the DocVariable field with Variable name and its type
paragraph.AppendField("FirstName", FieldType.FieldDocVariable);
paragraph = section.AddParagraph();
paragraph.AppendText("Last Name of the customer: ");
//Adds the DocVariable field with Variable name and its type
paragraph.AppendField("LastName", FieldType.FieldDocVariable);
//Adds the value for variable in WordDocument.Variable collection
document.Variables.Add("FirstName", "Jeff");
document.Variables.Add("LastName", "Smith");
//Updates the document fields
document.UpdateDocumentFields();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Cross reference

A cross-reference refers to an item that appears in another location in a document. You can create cross-reference to bookmarks in a document by using the `AppendCrossReference` method of `WParagraph` class.

Note: The Essential DocIO supports creating and updating the cross-reference fields only for bookmarks in a document.

The following code example explains how to append cross reference for bookmark in a Word document.

C#

```
//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
IWParagraph paragraph = section.AddParagraph();
//Adds text, bookmark start and end in the paragraph
paragraph.AppendBookmarkStart("Title");
paragraph.AppendText("Northwind Database");
paragraph.AppendBookmarkEnd("Title");
paragraph = section.AddParagraph();
paragraph.AppendText("The Northwind sample database (Northwind.mdb) is
included with all versions of Access. It provides data you can experiment
with and database objects that demonstrate features you might want to
implement in your own databases.");
section = document.AddSection();
section.AddParagraph();
paragraph = section.AddParagraph() as WParagraph;
//Gets the collection of bookmark start in the word document
List<Entity> items =
document.GetCrossReferenceItems(ReferenceType.Bookmark);
paragraph.AppendText("Bookmark Cross Reference starts here ");
//Appends the cross reference for bookmark "Title" with ContentText as
reference kind
paragraph.AppendCrossReference(ReferenceType.Bookmark,
ReferenceKind.ContentText, items[0], true, false, false, string.Empty);
//Updates the document Fields
document.UpdateDocumentFields();
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates an instance of a WordDocument
Dim document As New WordDocument()
Dim section As IWSection = document.AddSection()
Dim paragraph As IWParagraph = section.AddParagraph()
'Adds text, bookmark start and end in the paragraph
paragraph.AppendBookmarkStart("Title")
paragraph.AppendText("Northwind Database")
paragraph.AppendBookmarkEnd("Title")
paragraph = section.AddParagraph()
paragraph.AppendText("The Northwind sample database (Northwind.mdb) is
included with all versions of Access. It provides data you can experiment
with and database objects that demonstrate features you might want to
implement in your own databases.")
```



```

section = document.AddSection()
section.AddParagraph()
paragraph = TryCast(section.AddParagraph(), WParagraph)
'Gets the collection of bookmark start in the word document
Dim items As List(Of Entity) =
document.GetCrossReferenceItems(ReferenceType.Bookmark)
paragraph.AppendText("Bookmark Cross Reference starts here ")
'Appends the cross reference for bookmark "Title" with ContentText as
reference kind
paragraph.AppendCrossReference(ReferenceType.Bookmark,
ReferenceKind.ContentText, items(0), True, False, False, String.Empty)
'Updates the document Fields
document.UpdateDocumentFields()
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document
IWParagraph paragraph = section.AddParagraph();
//Adds text, bookmark start and end in the paragraph
paragraph.AppendBookmarkStart("Title");
paragraph.AppendText("Northwind Database");
paragraph.AppendBookmarkEnd("Title");
paragraph = section.AddParagraph();
paragraph.AppendText("The Northwind sample database (Northwind.mdb) is
included with all versions of Access. It provides data you can experiment
with and database objects that demonstrate features you might want to
implement in your own databases.");
section = document.AddSection();
section.AddParagraph();
paragraph = section.AddParagraph() as WParagraph;
//Gets the collection of bookmark start in the word document
List<Entity> items =
document.GetCrossReferenceItems(ReferenceType.Bookmark);
paragraph.AppendText("Bookmark Cross Reference starts here ");
//Appends the cross reference for bookmark "Title" with ContentText as
reference kind
paragraph.AppendCrossReference(ReferenceType.Bookmark,
ReferenceKind.ContentText, items[0], true, false, false, string.Empty);
//Updates the document Fields
document.UpdateDocumentFields();
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document
IWParagraph paragraph = section.AddParagraph();
//Adds text, bookmark start and end in the paragraph
paragraph.AppendBookmarkStart("Title");
paragraph.AppendText("Northwind Database");
paragraph.AppendBookmarkEnd("Title");
paragraph = section.AddParagraph();
paragraph.AppendText("The Northwind sample database (Northwind.mdb) is
included with all versions of Access. It provides data you can experiment
with and database objects that demonstrate features you might want to
implement in your own databases.");
section = document.AddSection();
section.AddParagraph();
paragraph = section.AddParagraph() as WParagraph;
//Gets the collection of bookmark start in the word document
List<Entity> items =
document.GetCrossReferenceItems(ReferenceType.Bookmark);
paragraph.AppendText("Bookmark Cross Reference starts here ");
//Appends the cross reference for bookmark "Title" with ContentText as
reference kind
paragraph.AppendCrossReference(ReferenceType.Bookmark,
ReferenceKind.ContentText, items[0], true, false, false, string.Empty);
//Updates the document Fields
document.UpdateDocumentFields();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document
IWParagraph paragraph = section.AddParagraph();
//Adds text, bookmark start and end in the paragraph
paragraph.AppendBookmarkStart("Title");
paragraph.AppendText("Northwind Database");
paragraph.AppendBookmarkEnd("Title");
paragraph = section.AddParagraph();
paragraph.AppendText("The Northwind sample database (Northwind.mdb) is
included with all versions of Access. It provides data you can experiment

```

```

with and database objects that demonstrate features you might want to
implement in your own databases.");
section = document.AddSection();
section.AddParagraph();
paragraph = section.AddParagraph() as WParagraph;
//Gets the collection of bookmark start in the word document
List<Entity> items =
document.GetCrossReferenceItems(ReferenceType.Bookmark);
paragraph.AppendText("Bookmark Cross Reference starts here ");
//Appends the cross reference for bookmark "Title" with ContentText as
reference kind
paragraph.AppendCrossReference(ReferenceType.Bookmark,
ReferenceKind.ContentText, items[0], true, false, false, string.Empty);
//Updates the document Fields
document.UpdateDocumentFields();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Unlink fields

You can replace the field with its most recent result in the Word document by unlinking the field using the **Unlink** API. When you unlink a field, its current result is converted to text or a graphic and can no longer be updated automatically.

The following code example shows how to unlink the fields in Word document.

C#

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Today's Date: ");
//Adds the new Date field in Word document with field name and its type
WField field = paragraph.AppendField("Date", FieldType.FieldDate) as WField;
//Updates the field
field.Update();
//Unlink the field
field.Unlink();
//Saves the document in the given name and format
document.Save("Sample.docx", FormatType.Docx);
//Releases the resources occupied by WordDocument instance
document.Close();

```

VB.NET

```

'Creates an instance of WordDocument class
Dim document As WordDocument = New WordDocument()
'Adds a new section into the Word Document
Dim section As IWSection = document.AddSection()
'Adds a new paragraph into Word document and appends text into paragraph
Dim paragraph As IWParagraph = section.AddParagraph()
paragraph.AppendText("Today's Date: ")
'Adds the new Date field in Word document with field name and its type
Dim field As WField = CType(paragraph.AppendField("Date",
FieldType.FieldDate), WField)
'Updates the field
field.Update()
'Unlink the field
field.Unlink()
'Saves the document in the given name and format
document.Save("Sample.docx", FormatType.Docx)
'Releases the resources occupied by WordDocument instance
document.Close()

```

UWP

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Today's Date: ");
//Adds the new Date field in Word document with field name and its type
WField field = paragraph.AppendField("Date", FieldType.FieldDate) as WField;
//Updates the field
field.Update();
//Unlink the field
field.Unlink();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Today's Date: ");

```

```
//Adds the new Date field in Word document with field name and its type
WField field = paragraph.AppendField("Date", FieldType.FieldDate) as WField;
//Updates the field
field.Update();
//Unlink the field
field.Unlink();
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
//Closes the Word document instance
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Creates an instance of WordDocument class
WordDocument document = new WordDocument();
//Adds a new section into the Word Document
IWSection section = document.AddSection();
//Adds a new paragraph into Word document and appends text into paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Today's Date: ");
//Adds the new Date field in Word document with field name and its type
WField field = paragraph.AppendField("Date", FieldType.FieldDate) as WField;
//Updates the field
field.Update();
//Unlink the field
field.Unlink();
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Note: Fields such as XE (Index Entry) fields and SEQ (Sequence) fields cannot be unlinked.

Working with Shapes

Shapes are drawing objects that include lines, curves, circles, rectangles, etc. It can be preset or custom geometry. You can create and manipulate the pre-defined shape in DOCX and WordML format documents.

Adding shapes

The following code example illustrates how to add pre-defined shape to the document.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Adds new shape to the document
Shape rectangle = paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150,
100);
//Sets position for shape
rectangle.VerticalPosition = 72;
rectangle.HorizontalPosition = 72;
paragraph = section.AddParagraph() as WParagraph;
//Adds textbody contents to the shape
paragraph = rectangle.TextBody.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("This text is in rounded rectangle
shape");
text.CharacterFormat.TextColor = Color.Green;
text.CharacterFormat.Bold = true;
//Adds another shape to the document
paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendBreak(BreakType.LineBreak);
Shape pentagon = paragraph.AppendShape(AutoShapeType.Pentagon, 100, 100);
paragraph = pentagon.TextBody.AddParagraph() as WParagraph;
paragraph.AppendText("This text is in pentagon shape");
pentagon.HorizontalPosition = 72;
pentagon.VerticalPosition = 200;
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As WParagraph = TryCast(section.AddParagraph(), WParagraph)
'Adds new shape to the document
Dim rectangle As Shape =
paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150, 100)
'Sets position for shape
rectangle.VerticalPosition = 72
rectangle.HorizontalPosition = 72
paragraph = TryCast(section.AddParagraph(), WParagraph)
'Adds textbody contents to the shape
paragraph = TryCast(rectangle.TextBody.AddParagraph(), WParagraph)
Dim text As IWTextRange = paragraph.AppendText("This text is in rounded
rectangle shape")
text.CharacterFormat.TextColor = Color.Green
text.CharacterFormat.Bold = True
'Adds another shape to the document
paragraph = TryCast(section.AddParagraph(), WParagraph)
paragraph.AppendBreak(BreakType.LineBreak)

```

```

Dim pentagon As Shape = paragraph.AppendShape(AutoShapeType.Pentagon, 100,
100)
paragraph = TryCast(pentagon.TextBody.AddParagraph(), WParagraph)
paragraph.AppendText("This text is in pentagon shape")
pentagon.HorizontalPosition = 72
pentagon.VerticalPosition = 200
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Adds new shape to the document
Shape rectangle = paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150,
100);
//Sets position for shape
rectangle.VerticalPosition = 72;
rectangle.HorizontalPosition = 72;
paragraph = section.AddParagraph() as WParagraph;
//Adds textbox contents to the shape
paragraph = rectangle.TextBody.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("This text is in rounded rectangle
shape");
text.CharacterFormat.TextColor = Color.Green;
text.CharacterFormat.Bold = true;
//Adds another shape to the document
paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendBreak(BreakType.LineBreak);
Shape pentagon = paragraph.AppendShape(AutoShapeType.Pentagon, 100, 100);
paragraph = pentagon.TextBody.AddParagraph() as WParagraph;
paragraph.AppendText("This text is in pentagon shape");
pentagon.HorizontalPosition = 72;
pentagon.VerticalPosition = 200;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document

```

```

IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Adds new shape to the document
Shape rectangle = paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150,
100);
//Sets position for shape
rectangle.VerticalPosition = 72;
rectangle.HorizontalPosition = 72;
paragraph = section.AddParagraph() as WParagraph;
//Adds textbody contents to the shape
paragraph = rectangle.TextBody.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("This text is in rounded rectangle
shape");
text.CharacterFormat.TextColor = Color.Green;
text.CharacterFormat.Bold = true;
//Adds another shape to the document
paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendBreak(BreakType.LineBreak);
Shape pentagon = paragraph.AppendShape(AutoShapeType.Pentagon, 100, 100);
paragraph = pentagon.TextBody.AddParagraph() as WParagraph;
paragraph.AppendText("This text is in pentagon shape");
pentagon.HorizontalPosition = 72;
pentagon.VerticalPosition = 200;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Adds new shape to the document
Shape rectangle = paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150,
100);
//Sets position for shape
rectangle.VerticalPosition = 72;
rectangle.HorizontalPosition = 72;
paragraph = section.AddParagraph() as WParagraph;
//Adds textbody contents to the shape
paragraph = rectangle.TextBody.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("This text is in rounded rectangle
shape");
text.CharacterFormat.TextColor = Syncfusion.Drawing.Color.Green;
text.CharacterFormat.Bold = true;
//Adds another shape to the document
paragraph = section.AddParagraph() as WParagraph;

```



```

paragraph.AppendBreak(BreakType.LineBreak);
Shape pentagon = paragraph.AppendShape(AutoShapeType.Pentagon, 100, 100);
paragraph = pentagon.TextBody.AddParagraph() as WParagraph;
paragraph.AppendText("This text is in pentagon shape");
pentagon.HorizontalPosition = 72;
pentagon.VerticalPosition = 200;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Format shapes

Shape can have formatting such as line color, fill color, positioning, wrap formats, etc. The following code example illustrates how to apply formatting options for shape.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
Shape rectangle = paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150,
100);
rectangle.VerticalPosition = 72;
rectangle.HorizontalPosition = 72;
paragraph = section.AddParagraph() as WParagraph;
paragraph = rectangle.TextBody.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("This text is in rounded rectangle
shape");
text.CharacterFormat.TextColor = Color.Green;
text.CharacterFormat.Bold = true;
//Applies fill color for shape
rectangle.FillFormat.Fill = true;
rectangle.FillFormat.Color = Color.LightGray;
//Applies wrap formats
rectangle.WrapFormat.TextWrappingStyle = TextWrappingStyle.Square;
rectangle.WrapFormat.TextWrappingType = TextWrappingType.Right;
//Sets horizontal and vertical origin
rectangle.HorizontalOrigin = HorizontalOrigin.Margin;
rectangle.VerticalOrigin = VerticalOrigin.Page;
//Sets line format
rectangle.LineFormat.DashStyle = LineDashing.Dot;
rectangle.LineFormat.Color = Color.DarkGray;
//Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As WParagraph = TryCast(section.AddParagraph(), WParagraph)
Dim rectangle As Shape =
paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150, 100)
rectangle.VerticalPosition = 72
rectangle.HorizontalPosition = 72
paragraph = TryCast(section.AddParagraph(), WParagraph)
paragraph = TryCast(rectangle.TextBody.AddParagraph(), WParagraph)
Dim text As ITextRange = paragraph.AppendText("This text is in rounded
rectangle shape")
text.CharacterFormat.TextColor = Color.Green
text.CharacterFormat.Bold = True
'Applies fill color for shape
rectangle.FillFormat.Fill = True
rectangle.FillFormat.Color = Color.LightGray
'Applies wrap formats
rectangle.WrapFormat.TextWrappingStyle = TextWrappingStyle.Square
rectangle.WrapFormat.TextWrappingType = TextWrappingType.Right
'Sets horizontal and vertical origin
rectangle.HorizontalOrigin = HorizontalOrigin.Margin
rectangle.VerticalOrigin = VerticalOrigin.Page
'Sets line format
rectangle.LineFormat.DashStyle = LineDashing.Dot
rectangle.LineFormat.Color = Color.DarkGray
'Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph() as WParagraph;
Shape rectangle = paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150,
100);
rectangle.VerticalPosition = 72;
rectangle.HorizontalPosition = 72;
paragraph = section.AddParagraph() as WParagraph;
paragraph = rectangle.TextBody.AddParagraph() as WParagraph;
ITextRange text = paragraph.AppendText("This text is in rounded rectangle
shape");
text.CharacterFormat.TextColor = Color.Green;
text.CharacterFormat.Bold = true;
//Applies fill color for shape
rectangle.FillFormat.Fill = true;
rectangle.FillFormat.Color = Color.LightGray;

```

```
//Applies wrap formats
rectangle.WrapFormat.TextWrappingStyle = TextWrappingStyle.Square;
rectangle.WrapFormat.TextWrappingType = TextWrappingType.Right;
//Sets horizontal and vertical origin
rectangle.HorizontalOrigin = HorizontalOrigin.Margin;
rectangle.VerticalOrigin = VerticalOrigin.Page;
//Sets line format
rectangle.LineFormat.DashStyle = LineDashing.Dot;
rectangle.LineFormat.Color = Color.DarkGray;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
document.Close();
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph() as WParagraph;
Shape rectangle = paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150, 100);
rectangle.VerticalPosition = 72;
rectangle.HorizontalPosition = 72;
paragraph = section.AddParagraph() as WParagraph;
paragraph = rectangle.TextBody.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("This text is in rounded rectangle shape");
text.CharacterFormat.TextColor = Color.Green;
text.CharacterFormat.Bold = true;
//Applies fill color for shape
rectangle.FillFormat.Fill = true;
rectangle.FillFormat.Color = Color.LightGray;
//Applies wrap formats
rectangle.WrapFormat.TextWrappingStyle = TextWrappingStyle.Square;
rectangle.WrapFormat.TextWrappingType = TextWrappingType.Right;
//Sets horizontal and vertical origin
rectangle.HorizontalOrigin = HorizontalOrigin.Margin;
rectangle.VerticalOrigin = VerticalOrigin.Page;
//Sets line format
rectangle.LineFormat.DashStyle = LineDashing.Dot;
rectangle.LineFormat.Color = Color.DarkGray;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
```

```
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph() as WParagraph;
Shape rectangle = paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150,
100);
rectangle.VerticalPosition = 72;
rectangle.HorizontalPosition = 72;
paragraph = section.AddParagraph() as WParagraph;
paragraph = rectangle.TextBody.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("This text is in rounded rectangle
shape");
text.CharacterFormat.TextColor = Syncfusion.Drawing.Color.Green;
text.CharacterFormat.Bold = true;
//Applies fill color for shape
rectangle.FillFormat.Fill = true;
rectangle.FillFormat.Color = Syncfusion.Drawing.Color.LightGray;
//Applies wrap formats
rectangle.WrapFormat.TextWrappingStyle = TextWrappingStyle.Square;
rectangle.WrapFormat.TextWrappingType = TextWrappingType.Right;
//Sets horizontal and vertical origin
rectangle.HorizontalOrigin = HorizontalOrigin.Margin;
rectangle.VerticalOrigin = VerticalOrigin.Page;
//Sets line format
rectangle.LineFormat.DashStyle = LineDashing.Dot;
rectangle.LineFormat.Color = Syncfusion.Drawing.Color.DarkGray;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Rotate shapes

You can rotate the shape and also apply flipping (horizontal and vertical) to it. The following code example explains how to rotate and flip the shape.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
```

```

IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
Shape rectangle = paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150,
100);
//Sets position for shape
rectangle.VerticalPosition = 72;
rectangle.HorizontalPosition = 72;
//Sets 90 degree rotation
rectangle.Rotation = 90;
//Sets horizontal flip
rectangle.FlipHorizontal = true;
paragraph = section.AddParagraph() as WParagraph;
paragraph = rectangle.TextBody.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("This text is in rounded rectangle
shape");
//Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As WParagraph = TryCast(section.AddParagraph(), WParagraph)
Dim rectangle As Shape =
paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150, 100)
'Sets position for shape
rectangle.VerticalPosition = 72
rectangle.HorizontalPosition = 72
'Sets 90 degree rotation
rectangle.Rotation = 90
'Sets horizontal flip
rectangle.FlipHorizontal = true
paragraph = TryCast(section.AddParagraph(), WParagraph)
paragraph = TryCast(rectangle.TextBody.AddParagraph(), WParagraph)
Dim text As IWTextRange = paragraph.AppendText("This text is in rounded
rectangle shape")
'Saves and closes the Word document
document.Save("Sample.docx", FormatType.Docx)
document.Close

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
Shape rectangle = paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150,
100);
//Sets position for shape

```

```

rectangle.VerticalPosition = 72;
rectangle.HorizontalPosition = 72;
//Sets 90 degree rotation
rectangle.Rotation = 90;
//Sets horizontal flip
rectangle.FlipHorizontal = true;
paragraph = section.AddParagraph() as WParagraph;
paragraph = rectangle.TextBody.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("This text is in rounded rectangle
shape");
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
Shape rectangle = paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150,
100);
//Sets position for shape
rectangle.VerticalPosition = 72;
rectangle.HorizontalPosition = 72;
//Sets 90 degree rotation
rectangle.Rotation = 90;
//Sets horizontal flip
rectangle.FlipHorizontal = true;
paragraph = section.AddParagraph() as WParagraph;
paragraph = rectangle.TextBody.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("This text is in rounded rectangle
shape");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document

```

```

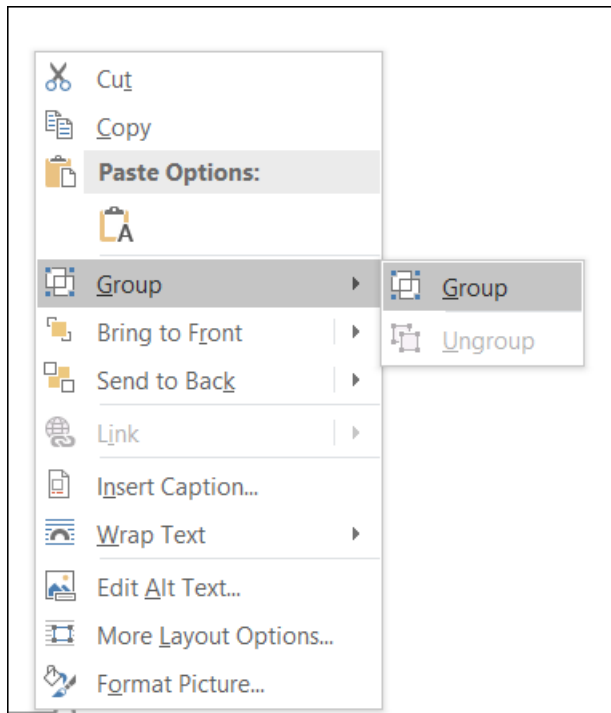
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
Shape rectangle = paragraph.AppendShape(AutoShapeType.RoundedRectangle, 150,
100);
//Sets position for shape
rectangle.VerticalPosition = 72;
rectangle.HorizontalPosition = 72;
//Sets 90 degree rotation
rectangle.Rotation = 90;
//Sets horizontal flip
rectangle.FlipHorizontal = true;
paragraph = section.AddParagraph() as WParagraph;
paragraph = rectangle.TextBody.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("This text is in rounded rectangle
shape");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Grouping shapes

Word library now allows you to create or group multiple shapes, pictures, text boxes, and charts as a group shape in Word document (DOCX) and preserve it as in DOCX and WordML format conversions.

You can create a document with group shapes by using Microsoft Word. It provides an option to group a set of shapes and images as a single shape and a group shape as individual item.



Key Features:

1. You can easily manage group of shapes, pictures, text boxes, or charts as a group shape.
2. You can move several shapes or images simultaneously and apply the same formatting properties for children of group shapes.

Note: 1. While grouping the shapes or other objects, the shapes should be positioned relative to the "Page".

2. While grouping the shapes or other objects, the wrapping style should not be "In Line with Text".

The following code example illustrates how to create group shape in Word document.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Creates new group shape
GroupShape groupShape = new GroupShape(document);
//Adds group shape to the paragraph.
paragraph.ChildEntities.Add(groupShape);
//Creates new shape
Shape shape = new Shape(document, AutoShapeType.RoundedRectangle);
//Sets height and width for shape
shape.Height = 100;
shape.Width = 150;
//Sets horizontal and vertical position
shape.HorizontalPosition = 72;
```



```

shape.VerticalPosition = 72;
//Set wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical origin
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
//Adds the specified shape to group shape
groupShape.Add(shape);
//Creates new picture
WPicture picture = new WPicture(document);
picture.LoadImage(Image.FromFile("Image.png"));
//Sets wrapping style for picture
picture.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets height and width for the image
picture.Height = 100;
picture.Width = 100;
//Sets horizontal and vertical position
picture.HorizontalPosition = 400;
picture.VerticalPosition = 150;
//Sets horizontal and vertical origin
picture.HorizontalOrigin = HorizontalOrigin.Page;
picture.VerticalOrigin = VerticalOrigin.Page;
//Adds the specified picture to group shape
groupShape.Add(picture);
//Creates new textbox
WTextBox textbox = new WTextBox(document);
textbox.TextBoxFormat.Width = 150;
textbox.TextBoxFormat.Height = 75;
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
textboxParagraph.AppendText("Text inside text box");
//Sets wrapping style for textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind;
//Sets horizontal and vertical position
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Adds the specified textbox to group shape
groupShape.Add(textbox);
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As WParagraph = TryCast(section.AddParagraph(), WParagraph)
'Creates new group shape
Dim groupShape As GroupShape = New GroupShape(document)

```

```

`Adds group shape to the paragraph
paragraph.ChildEntities.Add(groupShape)
`Creates new shape
Dim shape As Shape = New Shape(document, AutoShapeType.RoundedRectangle)
`Sets height and width for shape
shape.Height = 100
shape.Width = 150
`Sets horizontal and vertical position
shape.HorizontalPosition = 72
shape.VerticalPosition = 72
`Sets wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText
`Sets horizontal and vertical origin
shape.HorizontalOrigin = HorizontalOrigin.Page
shape.VerticalOrigin = VerticalOrigin.Page
`Adds the specified shape to group shape
groupShape.Add(shape)
`Creates new picture
Dim picture As WPicture = New WPicture(document)
picture.LoadImage(Image.FromFile("Image.png"))
`Sets wrapping style for picture
picture.TextWrappingStyle = TextWrappingStyle.InFrontOfText
`Sets height and width for the image
picture.Height = 100
picture.Width = 100
`Sets horizontal and vertical position
picture.HorizontalPosition = 400
picture.VerticalPosition = 150
`Sets horizontal and vertical origin
picture.HorizontalOrigin = HorizontalOrigin.Page
picture.VerticalOrigin = VerticalOrigin.Page
`Adds the specified picture to group shape
groupShape.Add(picture)
`Creates new textbox
Dim textbox As WTextBox = New WTextBox(document)
textbox.TextBoxFormat.Width = 150
textbox.TextBoxFormat.Height = 75
`Adds new text to the textbox body
Dim textboxParagraph As IWParagraph = textbox.TextBoxBody.AddParagraph()
textboxParagraph.AppendText("Text inside text box")
`Sets wrapping style for textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind
`Sets horizontal and vertical position
textbox.TextBoxFormat.HorizontalPosition = 200
textbox.TextBoxFormat.VerticalPosition = 200
`Sets horizontal and vertical origin
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page
`Adds the specified textbox to group shape
groupShape.Add(textbox)
`Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
`Closes the document
document.Close()

```

UWP

```

// "App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
// Creates a new Word document
WordDocument document = new WordDocument();
// Adds new section to the document
IWSection section = document.AddSection();
// Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
// Creates new group shape
GroupShape groupShape = new GroupShape(document);
// Adds group shape to the paragraph
paragraph.ChildEntities.Add(groupShape);
// Creates new shape
Shape shape = new Shape(document, AutoShapeType.RoundedRectangle);
// Sets height and width for shape
shape.Height = 100;
shape.Width = 150;
// Sets horizontal and vertical position
shape.HorizontalPosition = 72;
shape.VerticalPosition = 72;
// Set wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
// Sets horizontal and vertical origin
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
// Adds the specified shape to group shape
groupShape.Add(shape);
// Creates new picture
WPicture picture = new WPicture(document);
Stream imageStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Images.jpg");
picture.LoadImage(imageStream);
// Sets wrapping style for picture
picture.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
// Sets height and width for the image
picture.Height = 100;
picture.Width = 100;
// Sets horizontal and vertical position
picture.HorizontalPosition = 400;
picture.VerticalPosition = 150;
// Sets horizontal and vertical origin
picture.HorizontalOrigin = HorizontalOrigin.Page;
picture.VerticalOrigin = VerticalOrigin.Page;
// Adds the specified picture to group shape
groupShape.Add(picture);
// Creates new textbox
WTextBox textbox = new WTextBox(document);
textbox.TextBoxFormat.Width = 150;
textbox.TextBoxFormat.Height = 75;
// Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
textboxParagraph.AppendText("Text inside text box");
// Sets wrapping style for textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind;
// Sets horizontal and vertical position
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;

```

```
//Sets horizontal and vertical origin
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Adds the specified textbox to group shape
groupShape.Add(textbox);
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "GroupShape.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Creates new group shape
GroupShape groupShape = new GroupShape(document);
//Adds group shape to the paragraph
paragraph.ChildEntities.Add(groupShape);
//Creates new shape
Shape shape = new Shape(document, AutoShapeType.RoundedRectangle);
//Sets height and width for shape
shape.Height = 100;
shape.Width = 150;
//Sets horizontal and vertical position
shape.HorizontalPosition = 72;
shape.VerticalPosition = 72;
//Set wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical origin
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
//Adds the specified shape to group shape
groupShape.Add(shape);
//Creates new picture
WPicture picture = new WPicture(document);
FileStream imageStream = new FileStream("Image.png", FileMode.Open,
FileAccess.ReadWrite);
picture.LoadImage(imageStream);
//Sets wrapping style for picture
picture.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets height and width for the image
picture.Height = 100;
picture.Width = 100;
//Sets horizontal and vertical position
picture.HorizontalPosition = 400;
picture.VerticalPosition = 150;
```

```
//Sets horizontal and vertical origin
picture.HorizontalOrigin = HorizontalOrigin.Page;
picture.VerticalOrigin = VerticalOrigin.Page;
//Adds the specified picture to group shape
groupShape.Add(picture);
//Creates new textbox
WTextBox textbox = new WTextBox(document);
textbox.TextBoxFormat.Width = 150;
textbox.TextBoxFormat.Height = 75;
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
textboxParagraph.AppendText("Text inside text box");
//Sets wrapping style for textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind;
//Sets horizontal and vertical position
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Adds the specified textbox to group shape
groupShape.Add(textbox);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Creates new group shape
GroupShape groupShape = new GroupShape(document);
//Adds group shape to the paragraph.
paragraph.ChildEntities.Add(groupShape);
//Creates new shape
Shape shape = new Shape(document, AutoShapeType.RoundedRectangle);
//Sets height and width for shape
shape.Height = 100;
shape.Width = 150;
//Sets horizontal and vertical position
shape.HorizontalPosition = 72;
shape.VerticalPosition = 72;
//Set wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical origin
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
```

```

//Adds the specified shape to group shape
groupShape.Add(shape);
//Creates new picture
WPicture picture = new WPicture(document);
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream imageStream =
assembly.GetManifestResourceStream("WordToPDF.Assests.Image.png");
picture.LoadImage(imageStream);
//Sets wrapping style for picture
picture.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets height and width for the image
picture.Height = 100;
picture.Width = 100;
//Sets horizontal and vertical position
picture.HorizontalPosition = 400;
picture.VerticalPosition = 150;
//Sets horizontal and vertical origin
picture.HorizontalOrigin = HorizontalOrigin.Page;
picture.VerticalOrigin = VerticalOrigin.Page;
//Adds the specified picture to group shape
groupShape.Add(picture);
//Creates new textbox
WTextBox textbox = new WTextBox(document);
textbox.TextBoxFormat.Width = 150;
textbox.TextBoxFormat.Height = 75;
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
textboxParagraph.AppendText("Text inside text box");
//Sets wrapping style for textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind;
//Sets horizontal and vertical position
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Adds the specified textbox to group shape
groupShape.Add(textbox);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example illustrates how to add collection of shapes or images as a group shape in Word document.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Creates paragraph item collections to add child shapes
ParagraphItem[] paragraphItems = new ParagraphItem[3];
//Creates new shape
Shape shape = new Shape(document, AutoShapeType.RoundedRectangle);
//Sets height and width for shape
shape.Height = 100;
shape.Width = 150;
//Sets Wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical position for shape
shape.HorizontalPosition = 7;
shape.VerticalPosition = 72;
//Sets horizontal and vertical origin for shape
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
//Sets the shape as paragraph item
paragraphItems[0] = shape;
//Appends new textbox to the document
WTextBox textbox = new WTextBox(document);
//Sets height and width for textbox
textbox.TextBoxFormat.Width = 150;
textbox.TextBoxFormat.Height = 75;
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
//Adds new text to the textbox paragraph
textboxParagraph.AppendText("Text inside text box");
//Sets wrapping style for textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind;
//Sets horizontal and vertical position for textbox
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin for textbox
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Sets the textbox as paragraph item
paragraphItems[1] = textbox;
//Appends new chart to the document
WChart chart = new WChart(document);
//Sets height and width for chart
chart.Height = 270;
chart.Width = 446;
//Sets wrapping style for chart
chart.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets chart type
chart.ChartType = OfficeChartType.Pie;
chart.VerticalPosition = 350;
//Sets chart title
chart.ChartTitle = "Best Selling Products";
//Sets font and size for chart title
chart.ChartTitleArea.FontName = "Calibri";
chart.ChartTitleArea.Size = 14;

```

```

//Sets data for chart
chart.ChartData.SetValue(1, 1, "");
chart.ChartData.SetValue(1, 2, "Sales");
chart.ChartData.SetValue(2, 1, "Phyllis Lapin");
chart.ChartData.SetValue(2, 2, 141.396);
chart.ChartData.SetValue(3, 1, "Stanley Hudson");
chart.ChartData.SetValue(3, 2, 80.368);
chart.ChartData.SetValue(4, 1, "Bernard Shah");
chart.ChartData.SetValue(4, 2, 71.155);
chart.ChartData.SetValue(5, 1, "Patricia Lincoln");
chart.ChartData.SetValue(5, 2, 47.234);
chart.ChartData.SetValue(6, 1, "Camembert Pierrot");
chart.ChartData.SetValue(6, 2, 46.825);
chart.ChartData.SetValue(7, 1, "Thomas Hardy");
chart.ChartData.SetValue(7, 2, 42.593);
chart.ChartData.SetValue(8, 1, "Hanna Moos");
chart.ChartData.SetValue(8, 2, 41.819);
chart.ChartData.SetValue(9, 1, "Alice Mutton");
chart.ChartData.SetValue(9, 2, 32.698);
chart.ChartData.SetValue(10, 1, "Christina Berglund");
chart.ChartData.SetValue(10, 2, 29.171);
chart.ChartData.SetValue(11, 1, "Elizabeth Lincoln");
chart.ChartData.SetValue(11, 2, 25.696);
//Creates a new chart series with the name "Sales"
IOfficeChartSerie pieSeries = chart.Series.Add("Sales");
//Sets value for the chart series
pieSeries.Values = chart.ChartData[2, 2, 11, 2];
//Sets data label
pieSeries.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
pieSeries.DataPoints.DefaultDataPoint.DataLabels.Position =
OfficeDataLabelPosition.Outside;
//Sets background color
chart.ChartArea.Fill.ForeColor = Color.FromArgb(242, 242, 242);
chart.PlotArea.Fill.ForeColor = Color.FromArgb(242, 242, 242);
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Sets category labels
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData[2, 1, 11, 1];
//Sets the chart as paragraph item
paragraphItems[2] = chart;
//Creates new group shape
GroupShape groupShape = new GroupShape(document, paragraphItems);
groupShape.HorizontalPosition = 72;
//Adds the group shape to the paragraph
paragraph.ChildEntities.Add(groupShape);
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section

```



```

Dim paragraph As WParagraph = TryCast(section.AddParagraph(), WParagraph)
'Creates paragraph item collections to add child shapes
Dim paragraphItems As ParagraphItem() = New ParagraphItem(2) {}
'Creates new shape
Dim shape As Shape = New Shape(document, AutoShapeType.RoundedRectangle)
'Sets height and width for shape
shape.Height = 100
shape.Width = 150
'Sets Wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText
'Sets horizontal and vertical position for shape
shape.HorizontalPosition = 7
shape.VerticalPosition = 72
'Sets horizontal and vertical origin for shape
shape.HorizontalOrigin = HorizontalOrigin.Page
shape.VerticalOrigin = VerticalOrigin.Page
'Sets the shape as paragraph item
paragraphItems(0) = shape
'Appends new textbox to the document
Dim textbox As WTextBox = New WTextBox(document)
'Sets height and width for textbox
textbox.TextBoxFormat.Width = 150
textbox.TextBoxFormat.Height = 75
'Adds new text to the textbox body
Dim textboxParagraph As IWParagraph = textbox.TextBoxBody.AddParagraph()
'Adds new text to the textbox paragraph
textboxParagraph.AppendText("Text inside text box")
'Sets wrapping style for textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind
'Sets horizontal and vertical position for textbox
textbox.TextBoxFormat.HorizontalPosition = 200
textbox.TextBoxFormat.VerticalPosition = 200
'Sets horizontal and vertical origin for textbox
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page
'Sets the textbox as paragraph item
paragraphItems(1) = textbox
'Appends new chart to the document
Dim chart As WChart = New WChart(document)
'Sets height and width for chart
chart.Height = 270
chart.Width = 446
'Sets wrapping style for chart
chart.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText
'Sets chart type
chart.ChartType = OfficeChartType.Pie
chart.VerticalPosition = 350
'Sets chart title
'Sets font and size for chart title
chart.ChartTitle = "Best Selling Products"
chart.ChartTitleArea.FontName = "Calibri"
chart.ChartTitleArea.Size = 14
'Sets data for chart
chart.ChartData.SetValue(1, 1, "")
chart.ChartData.SetValue(1, 2, "Sales")
chart.ChartData.SetValue(2, 1, "Phyllis Lapin")
chart.ChartData.SetValue(2, 2, 141.396)

```

```

chart.ChartData.SetValue(3, 1, "Stanley Hudson")
chart.ChartData.SetValue(3, 2, 80.368)
chart.ChartData.SetValue(4, 1, "Bernard Shah")
chart.ChartData.SetValue(4, 2, 71.155)
chart.ChartData.SetValue(5, 1, "Patricia Lincoln")
chart.ChartData.SetValue(5, 2, 47.234)
chart.ChartData.SetValue(6, 1, "Camembert Pierrot")
chart.ChartData.SetValue(6, 2, 46.825)
chart.ChartData.SetValue(7, 1, "Thomas Hardy")
chart.ChartData.SetValue(7, 2, 42.593)
chart.ChartData.SetValue(8, 1, "Hanna Moos")
chart.ChartData.SetValue(8, 2, 41.819)
chart.ChartData.SetValue(9, 1, "Alice Mutton")
chart.ChartData.SetValue(9, 2, 32.698)
chart.ChartData.SetValue(10, 1, "Christina Berglund")
chart.ChartData.SetValue(10, 2, 29.171)
chart.ChartData.SetValue(11, 1, "Elizabeth Lincoln")
chart.ChartData.SetValue(11, 2, 25.696)
`Creates a new chart series with the name "Sales"
Dim pieSeries As IOOfficeChartSerie = chart.Series.Add("Sales")
`Sets value for the chart series
pieSeries.Values = chart.ChartData(2, 2, 11, 2)
`Sets data label
pieSeries.DataPoints.DefaultDataPoint.DataLabels.IsValue = True
pieSeries.DataPoints.DefaultDataPoint.DataLabels.Position =
OfficeDataLabelPosition.Outside
`Sets background color
chart.ChartArea.Fill.ForeColor = Color.FromArgb(242, 242, 242)
chart.PlotArea.Fill.ForeColor = Color.FromArgb(242, 242, 242)
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None
`Sets category labels
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData(2, 1, 11, 1)
`Sets the chart as paragraph item
paragraphItems(2) = chart
`Creates new group shape
Dim groupShape As GroupShape = New GroupShape(document, paragraphItems)
groupShape.HorizontalPosition = 72
`Adds the group shape to the paragraph
paragraph.ChildEntities.Add(groupShape)
`Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
`Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Creates paragraph item collections to add child shapes
ParagraphItem[] paragraphItems = new ParagraphItem[3];
//Creates new shape
Shape shape = new Shape(document, AutoShapeType.RoundedRectangle);

```

```

//Sets height and width for shape
shape.Height = 100;
shape.Width = 150;
//Sets Wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical position for shape
shape.HorizontalPosition = 7;
shape.VerticalPosition = 72;
//Sets horizontal and vertical origin for shape
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
//Sets the shape as paragraph item
paragraphItems[0] = shape;
//Appends new textbox to the document
WTextBox textbox = new WTextBox(document);
//Sets height and width for textbox
textbox.TextBoxFormat.Width = 150;
textbox.TextBoxFormat.Height = 75;
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
//Adds new text to the textbox paragraph
textboxParagraph.AppendText("Text inside text box");
//Sets wrapping style for textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind;
//Sets horizontal and vertical position for textbox
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin for textbox
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Sets the textbox as paragraph item
paragraphItems[1] = textbox;
//Appends new chart to the document
WChart chart = new WChart(document);
//Sets height and width for chart
chart.Height = 270;
chart.Width = 446;
//Sets wrapping style for chart
chart.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets chart type
chart.ChartType = OfficeChartType.Pie;
chart.VerticalPosition = 350;
//Sets chart title
chart.ChartTitle = "Best Selling Products";
//Sets font and size for chart title
chart.ChartTitleArea.FontName = "Calibri";
chart.ChartTitleArea.Size = 14;
//Sets data for chart
chart.ChartData.SetValue(1, 1, "");
chart.ChartData.SetValue(1, 2, "Sales");
chart.ChartData.SetValue(2, 1, "Phyllis Lapin");
chart.ChartData.SetValue(2, 2, 141.396);
chart.ChartData.SetValue(3, 1, "Stanley Hudson");
chart.ChartData.SetValue(3, 2, 80.368);
chart.ChartData.SetValue(4, 1, "Bernard Shah");
chart.ChartData.SetValue(4, 2, 71.155);
chart.ChartData.SetValue(5, 1, "Patricia Lincoln");

```

```

chart.ChartData.SetValue(5, 2, 47.234);
chart.ChartData.SetValue(6, 1, "Camembert Pierrot");
chart.ChartData.SetValue(6, 2, 46.825);
chart.ChartData.SetValue(7, 1, "Thomas Hardy");
chart.ChartData.SetValue(7, 2, 42.593);
chart.ChartData.SetValue(8, 1, "Hanna Moos");
chart.ChartData.SetValue(8, 2, 41.819);
chart.ChartData.SetValue(9, 1, "Alice Mutton");
chart.ChartData.SetValue(9, 2, 32.698);
chart.ChartData.SetValue(10, 1, "Christina Berglund");
chart.ChartData.SetValue(10, 2, 29.171);
chart.ChartData.SetValue(11, 1, "Elizabeth Lincoln");
chart.ChartData.SetValue(11, 2, 25.696);
//Creates a new chart series with the name "Sales"
IOfficeChartSerie pieSeries = chart.Series.Add("Sales");
//Sets value for the chart series
pieSeries.Values = chart.ChartData[2, 2, 11, 2];
//Sets data label
pieSeries.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
pieSeries.DataPoints.DefaultDataPoint.DataLabels.Position =
OfficeDataLabelPosition.Outside;
//Sets background color
chart.ChartArea.Fill.ForeColor = Syncfusion.Drawing.Color.FromArgb(242, 242,
242);
chart.PlotArea.Fill.ForeColor = Syncfusion.Drawing.Color.FromArgb(242, 242,
242);
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Sets category labels
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData[2, 1, 11, 1];
paragraphItems[2] = chart;
//Creates new group shape
GroupShape groupShape = new GroupShape(document, paragraphItems);
groupShape.HorizontalPosition = 72;
//Adds the group shape to the paragraph
paragraph.ChildEntities.Add(groupShape);
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "GroupSahpe.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Creates paragraph item collections to add child shapes
ParagraphItem[] paragraphItems = new ParagraphItem[3];

```

```

//Creates new shape
Shape shape = new Shape(document, AutoShapeType.RoundedRectangle);
//Sets height and width for shape
shape.Height = 100;
shape.Width = 150;
//Sets Wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical position for shape
shape.HorizontalPosition = 7;
shape.VerticalPosition = 72;
//Sets horizontal and vertical origin for shape
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
//Sets the shape as paragraph item
paragraphItems[0] = shape;
//Appends new textbox to the document
WTextBox textbox = new WTextBox(document);
//Sets height and width for textbox
textbox.TextBoxFormat.Width = 150;
textbox.TextBoxFormat.Height = 75;
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
//Adds new text to the textbox paragraph
textboxParagraph.AppendText("Text inside text box");
//Sets wrapping style for textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind;
//Sets horizontal and vertical position for textbox
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin for textbox
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Sets the textbox as paragraph item
paragraphItems[1] = textbox;
//Appends new chart to the document
WChart chart = new WChart(document);
//Sets height and width for chart
chart.Height = 270;
chart.Width = 446;
//Sets wrapping style for chart
chart.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets chart type
chart.ChartType = OfficeChartType.Pie;
chart.VerticalPosition = 350;
//Sets chart title
chart.ChartTitle = "Best Selling Products";
//Sets font and size for chart title
chart.ChartTitleArea.FontName = "Calibri";
chart.ChartTitleArea.Size = 14;
//Sets data for chart
chart.ChartData.SetValue(1, 1, "");
chart.ChartData.SetValue(1, 2, "Sales");
chart.ChartData.SetValue(2, 1, "Phyllis Lapin");
chart.ChartData.SetValue(2, 2, 141.396);
chart.ChartData.SetValue(3, 1, "Stanley Hudson");
chart.ChartData.SetValue(3, 2, 80.368);
chart.ChartData.SetValue(4, 1, "Bernard Shah");

```

```

chart.ChartData.SetValue(4, 2, 71.155);
chart.ChartData.SetValue(5, 1, "Patricia Lincoln");
chart.ChartData.SetValue(5, 2, 47.234);
chart.ChartData.SetValue(6, 1, "Camembert Pierrot");
chart.ChartData.SetValue(6, 2, 46.825);
chart.ChartData.SetValue(7, 1, "Thomas Hardy");
chart.ChartData.SetValue(7, 2, 42.593);
chart.ChartData.SetValue(8, 1, "Hanna Moos");
chart.ChartData.SetValue(8, 2, 41.819);
chart.ChartData.SetValue(9, 1, "Alice Mutton");
chart.ChartData.SetValue(9, 2, 32.698);
chart.ChartData.SetValue(10, 1, "Christina Berglund");
chart.ChartData.SetValue(10, 2, 29.171);
chart.ChartData.SetValue(11, 1, "Elizabeth Lincoln");
chart.ChartData.SetValue(11, 2, 25.696);
//Creates a new chart series with the name "Sales"
IOfficeChartSerie pieSeries = chart.Series.Add("Sales");
//Sets value for the chart series
pieSeries.Values = chart.ChartData[2, 2, 11, 2];
//Sets data label
pieSeries.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
pieSeries.DataPoints.DefaultDataPoint.DataLabels.Position =
OfficeDataLabelPosition.Outside;
//Sets background color
chart.ChartArea.Fill.ForeColor = Color.FromArgb(242, 242, 242);
chart.PlotArea.Fill.ForeColor = Color.FromArgb(242, 242, 242);
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Sets category labels
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData[2, 1, 11, 1];
//Sets the chart as paragraph item
paragraphItems[2] = chart;
//Creates new group shape
GroupShape groupShape = new GroupShape(document, paragraphItems);
groupShape.HorizontalPosition = 72;
//Adds the group shape to the paragraph
paragraph.ChildEntities.Add(groupShape);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Creates paragraph item collections to add child shapes
ParagraphItem[] paragraphItems = new ParagraphItem[3];
//Creates new shape

```

```

Shape shape = new Shape(document, AutoShapeType.RoundedRectangle);
//Sets height and width for shape
shape.Height = 100;
shape.Width = 150;
//Sets Wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical position for shape
shape.HorizontalPosition = 7;
shape.VerticalPosition = 72;
//Sets horizontal and vertical origin for shape
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
//Sets the shape as paragraph item
paragraphItems[0] = shape;
//Appends new textbox to the document
WTextBox textbox = new WTextBox(document);
//Sets height and width for textbox
textbox.TextBoxFormat.Width = 150;
textbox.TextBoxFormat.Height = 75;
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
//Adds new text to the textbox paragraph
textboxParagraph.AppendText("Text inside text box");
//Sets wrapping style for textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind;
//Sets horizontal and vertical position for textbox
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin for textbox
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Sets the textbox as paragraph item
paragraphItems[1] = textbox;
//Appends new chart to the document
WChart chart = new WChart(document);
//Sets height and width for chart
chart.Height = 270;
chart.Width = 446;
//Sets wrapping style for chart
chart.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets chart type
chart.ChartType = OfficeChartType.Pie;
chart.VerticalPosition = 350;
//Sets chart title
chart.ChartTitle = "Best Selling Products";
//Sets font and size for chart title
chart.ChartTitleArea.FontName = "Calibri";
chart.ChartTitleArea.Size = 14;
//Sets data for chart
chart.ChartData.SetValue(1, 1, "");
chart.ChartData.SetValue(1, 2, "Sales");
chart.ChartData.SetValue(2, 1, "Phyllis Lapin");
chart.ChartData.SetValue(2, 2, 141.396);
chart.ChartData.SetValue(3, 1, "Stanley Hudson");
chart.ChartData.SetValue(3, 2, 80.368);
chart.ChartData.SetValue(4, 1, "Bernard Shah");
chart.ChartData.SetValue(4, 2, 71.155);

```

```

chart.ChartData.SetValue(5, 1, "Patricia Lincoln");
chart.ChartData.SetValue(5, 2, 47.234);
chart.ChartData.SetValue(6, 1, "Camembert Pierrot");
chart.ChartData.SetValue(6, 2, 46.825);
chart.ChartData.SetValue(7, 1, "Thomas Hardy");
chart.ChartData.SetValue(7, 2, 42.593);
chart.ChartData.SetValue(8, 1, "Hanna Moos");
chart.ChartData.SetValue(8, 2, 41.819);
chart.ChartData.SetValue(9, 1, "Alice Mutton");
chart.ChartData.SetValue(9, 2, 32.698);
chart.ChartData.SetValue(10, 1, "Christina Berglund");
chart.ChartData.SetValue(10, 2, 29.171);
chart.ChartData.SetValue(11, 1, "Elizabeth Lincoln");
chart.ChartData.SetValue(11, 2, 25.696);
//Creates a new chart series with the name "Sales"
IOfficeChartSerie pieSeries = chart.Series.Add("Sales");
//Sets value for the chart series
pieSeries.Values = chart.ChartData[2, 2, 11, 2];
//Sets data label
pieSeries.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
pieSeries.DataPoints.DefaultDataPoint.DataLabels.Position =
OfficeDataLabelPosition.Outside;
//Sets background color
chart.ChartArea.Fill.ForeColor = Syncfusion.Drawing.Color.FromArgb(242, 242,
242);
chart.PlotArea.Fill.ForeColor = Syncfusion.Drawing.Color.FromArgb(242, 242,
242);
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Sets category labels
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData[2, 1, 11, 1];
paragraphItems[2] = chart;
//Creates new group shape
GroupShape groupShape = new GroupShape(document, paragraphItems);
groupShape.HorizontalPosition = 72;
//Adds the group shape to the paragraph
paragraph.ChildEntities.Add(groupShape);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Nested group shapes

The following code example illustrates how to group the nested group shapes as a group shape in Word document.

C#

```
//Creates a new Word document
```



```
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Creates new group shape
GroupShape groupShape = new GroupShape(document);
//Adds group shape to the paragraph
paragraph.ChildEntities.Add(groupShape);
//Appends new shape to the document
Shape shape = new Shape(document, AutoShapeType.RoundedRectangle);
//Sets height and width for shape
shape.Height = 100;
shape.Width = 150;
//Sets Wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical position for shape
shape.HorizontalPosition = 72;
shape.VerticalPosition = 72;
//Sets horizontal and vertical origin for shape
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
//Adds the specified shape to group shape
groupShape.Add(shape);
//Appends new picture to the document
WPicture picture = new WPicture(document);
//Loads image from the file
picture.LoadImage(Image.FromFile("Image.png"));
//Sets wrapping style for picture
picture.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets height and width for the picture
picture.Height = 100;
picture.Width = 100;
//Sets horizontal and vertical position for the picture
picture.HorizontalPosition = 400;
picture.VerticalPosition = 150;
//Sets horizontal and vertical origin for the picture
picture.HorizontalOrigin = HorizontalOrigin.Page;
picture.VerticalOrigin = VerticalOrigin.Page;
//Adds specified picture to the group shape
groupShape.Add(picture);
//Creates new nested group shape
GroupShape nestedGroupShape = new GroupShape(document);
//Appends new textbox to the document
WTextBox textbox = new WTextBox(document);
//Sets width and height for the textbox
textbox.TextBoxFormat.Width = 150;
textbox.TextBoxFormat.Height = 75;
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
//Adds new text to the textbox paragraph
textboxParagraph.AppendText("Text inside text box");
//Sets wrapping style for the textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind;
//Sets horizontal and vertical position for the textbox
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
```

```

//Sets horizontal and vertical origin for the textbox
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Adds specified textbox to the nested group shape
nestedGroupShape.Add(textbox);
//Appends new shape to the document
shape = new Shape(document, AutoShapeType.Oval);
//Sets height and width for the new shape
shape.Height = 100;
shape.Width = 150;
//Sets horizontal and vertical position for the shape
shape.HorizontalPosition = 200;
shape.VerticalPosition = 72;
//Sets horizontal and vertical origin for the shape
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
//Sets horizontal and vertical position for the nested group shape
nestedGroupShape.HorizontalPosition = 72;
nestedGroupShape.VerticalPosition = 72;
//Adds specified shape to the nested group shape
nestedGroupShape.Add(shape);
//Adds nested group shape to the group shape of the paragraph
groupShape.Add(nestedGroupShape);
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

`Creates a new Word document
Dim document As WordDocument = New WordDocument()
`Adds new section to the document
Dim section As IWSection = document.AddSection()
`Adds new paragraph to the section
Dim paragraph As WParagraph = TryCast(section.AddParagraph(), WParagraph)
`Creates new group shape
Dim groupShape As GroupShape = New GroupShape(document)
`Adds group shape to the paragraph
paragraph.ChildEntities.Add(groupShape)
`Appends new shape to the document
Dim shape As Shape = New Shape(document, AutoShapeType.RoundedRectangle)
`Sets height and width for shape
shape.Height = 100
shape.Width = 150
`Sets Wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText
`Sets horizontal and vertical position for shape
shape.HorizontalPosition = 72
shape.VerticalPosition = 72
`Sets horizontal and vertical origin for shape
shape.HorizontalOrigin = HorizontalOrigin.Page
shape.VerticalOrigin = VerticalOrigin.Page
`Adds the specified shape to group shape
groupShape.Add(shape)
`Appends new picture to the document

```

```

Dim picture As WPicture = New WPicture(document)
'Loads image from the file
picture.LoadImage(Image.FromFile("Image.jpg"))
'Sets wrapping style for picture
picture.TextWrappingStyle = TextWrappingStyle.InFrontOfText
'Sets height and width for the picture
picture.Height = 100
picture.Width = 100
'Sets horizontal and vertical position for the picture
picture.HorizontalPosition = 400
picture.VerticalPosition = 150
'Sets horizontal and vertical origin for the picture
picture.HorizontalOrigin = HorizontalOrigin.Page
picture.VerticalOrigin = VerticalOrigin.Page
'Adds specified picture to the group shape
groupShape.Add(picture)
'Creates new nested group shape
Dim nestedGroupShape As GroupShape = New GroupShape(document)
'Appends new textbox to the document
Dim textbox As WTextBox = New WTextBox(document)
'Sets width and height for the textbox
textbox.TextBoxFormat.Width = 150
textbox.TextBoxFormat.Height = 75
'Adds new text to the textbox body
Dim textboxParagraph As IWParagraph = textbox.TextBoxBody.AddParagraph()
'Adds new text to the textbox paragraph
textboxParagraph.AppendText("Text inside text box")
'Sets wrapping style for the textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind
'Sets horizontal and vertical position for the textbox
textbox.TextBoxFormat.HorizontalPosition = 200
textbox.TextBoxFormat.VerticalPosition = 200
'Sets horizontal and vertical origin for the textbox
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page
'Adds specified textbox to the nested group shape
nestedGroupShape.Add(textbox)
'Appends new shape to the document
shape = New Shape(document, AutoShapeType.Oval)
'Sets height and width for the new shape
shape.Height = 100
shape.Width = 150
'Sets horizontal and vertical position for the shape
shape.HorizontalPosition = 200
shape.VerticalPosition = 72
'Sets horizontal and vertical origin for the shape
shape.HorizontalOrigin = HorizontalOrigin.Page
shape.VerticalOrigin = VerticalOrigin.Page
'Sets horizontal and vertical position for the nested group shape
nestedGroupShape.HorizontalPosition = 72
nestedGroupShape.VerticalPosition = 72
'Adds specified shape to the nested group shape
nestedGroupShape.Add(shape)
'Adds nested group shape to the group shape of the paragraph
groupShape.Add(nestedGroupShape)
groupShape.HorizontalPosition = 142
'Saves the Word document

```

```
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Creates new group shape
GroupShape groupShape = new GroupShape(document);
//Adds group shape to the paragraph
paragraph.ChildEntities.Add(groupShape);
//Appends new shape to the document
Shape shape = new Shape(document, AutoShapeType.RoundedRectangle);
//Sets height and width for shape
shape.Height = 100;
shape.Width = 150;
//Sets Wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical position for shape
shape.HorizontalPosition = 72;
shape.VerticalPosition = 72;
//Sets horizontal and vertical origin for shape
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
//Adds the specified shape to group shape
groupShape.Add(shape);
//Appends new picture to the document
WPicture picture = new WPicture(document);
Stream imageStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Dummy-
Images.jpg");
//Loads image from the file
picture.LoadImage(imageStream);
//Sets wrapping style for picture
picture.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets height and width for the picture
picture.Height = 100;
picture.Width = 100;
//Sets horizontal and vertical position for the picture
picture.HorizontalPosition = 400;
picture.VerticalPosition = 150;
//Sets horizontal and vertical origin for the picture
picture.HorizontalOrigin = HorizontalOrigin.Page;
picture.VerticalOrigin = VerticalOrigin.Page;
//Adds specified picture to the group shape
groupShape.Add(picture);
//Creates new nested group shape
GroupShape nestedGroupShape = new GroupShape(document);
//Appends new textbox to the document
```

```

WTextBox textbox = new WTextBox(document);
//Sets width and height for the textbox
textbox.TextBoxFormat.Width = 150;
textbox.TextBoxFormat.Height = 75;
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
//Adds new text to the textbox paragraph
textboxParagraph.AppendText("Text inside text box");
//Sets wrapping style for the textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind;
//Sets horizontal and vertical position for the textbox
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin for the textbox
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Adds specified textbox to the nested group shape
nestedGroupShape.Add(textbox);
//Appends new shape to the document
shape = new Shape(document, AutoShapeType.Oval);
//Sets height and width for the new shape
shape.Height = 100;
shape.Width = 150;
//Sets horizontal and vertical position for the shape
shape.HorizontalPosition = 200;
shape.VerticalPosition = 72;
//Sets horizontal and vertical origin for the shape
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
//Sets horizontal and vertical position for the nested group shape
nestedGroupShape.HorizontalPosition = 72;
nestedGroupShape.VerticalPosition = 72;
//Adds specified shape to the nested group shape
nestedGroupShape.Add(shape);
//Adds nested group shape to the group shape of the paragraph
groupShape.Add(nestedGroupShape);
groupShape.HorizontalPosition = 142;
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "GroupShape.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;

```

```
//Creates new group shape
GroupShape groupShape = new GroupShape(document);
//Adds group shape to the paragraph
paragraph.ChildEntities.Add(groupShape);
//Appends new shape to the document
Shape shape = new Shape(document, AutoShapeType.RoundedRectangle);
//Sets height and width for shape
shape.Height = 100;
shape.Width = 150;
//Sets Wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical position for shape
shape.HorizontalPosition = 72;
shape.VerticalPosition = 72;
//Sets horizontal and vertical origin for shape
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
//Adds the specified shape to group shape
groupShape.Add(shape);
//Appends new picture to the document
WPicture picture = new WPicture(document);
//Loads image from the file
FileStream imageStream = new FileStream("Image.png", FileMode.Open,
FileAccess.ReadWrite);
picture.LoadImage(imageStream);
//Sets wrapping style for picture
picture.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets height and width for the picture
picture.Height = 100;
picture.Width = 100;
//Sets horizontal and vertical position for the picture
picture.HorizontalPosition = 400;
picture.VerticalPosition = 150;
//Sets horizontal and vertical origin for the picture
picture.HorizontalOrigin = HorizontalOrigin.Page;
picture.VerticalOrigin = VerticalOrigin.Page;
//Adds specified picture to the group shape
groupShape.Add(picture);
//Creates new nested group shape
GroupShape nestedGroupShape = new GroupShape(document);
//Appends new textbox to the document
WTextBox textbox = new WTextBox(document);
//Sets width and height for the textbox
textbox.TextBoxFormat.Width = 150;
textbox.TextBoxFormat.Height = 75;
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
//Adds new text to the textbox paragraph
textboxParagraph.AppendText("Text inside text box");
//Sets wrapping style for the textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind;
//Sets horizontal and vertical position for the textbox
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin for the textbox
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
```

```
//Adds specified textbox to the nested group shape
nestedGroupShape.Add(textbox);
//Appends new shape to the document
shape = new Shape(document, AutoShapeType.Oval);
//Sets height and width for the new shape
shape.Height = 100;
shape.Width = 150;
//Sets horizontal and vertical position for the shape
shape.HorizontalPosition = 200;
shape.VerticalPosition = 72;
//Sets horizontal and vertical origin for the shape
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
//Sets horizontal and vertical position for the nested group shape
nestedGroupShape.HorizontalPosition = 72;
nestedGroupShape.VerticalPosition = 72;
//Adds specified shape to the nested group shape
nestedGroupShape.Add(shape);
//Adds nested group shape to the group shape of the paragraph
groupShape.Add(nestedGroupShape);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Creates new group shape
GroupShape groupShape = new GroupShape(document);
//Adds group shape to the paragraph
paragraph.ChildEntities.Add(groupShape);
//Appends new shape to the document
Shape shape = new Shape(document, AutoShapeType.RoundedRectangle);
//Sets height and width for shape
shape.Height = 100;
shape.Width = 150;
//Sets Wrapping style for shape
shape.WrapFormat.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets horizontal and vertical position for shape
shape.HorizontalPosition = 72;
shape.VerticalPosition = 72;
//Sets horizontal and vertical origin for shape
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
```

```

//Adds the specified shape to group shape
groupShape.Add(shape);
//Appends new picture to the document
WPicture picture = new WPicture(document);
Stream imageStream =
assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.Dummy-
Images.jpg");
//Loads image from the file
picture.LoadImage(imageStream);
//Sets wrapping style for picture
picture.TextWrappingStyle = TextWrappingStyle.InFrontOfText;
//Sets height and width for the picture
picture.Height = 100;
picture.Width = 100;
//Sets horizontal and vertical position for the picture
picture.HorizontalPosition = 400;
picture.VerticalPosition = 150;
//Sets horizontal and vertical origin for the picture
picture.HorizontalOrigin = HorizontalOrigin.Page;
picture.VerticalOrigin = VerticalOrigin.Page;
//Adds specified picture to the group shape
groupShape.Add(picture);
//Creates new nested group shape
GroupShape nestedGroupShape = new GroupShape(document);
//Appends new textbox to the document
WTextBox textbox = new WTextBox(document);
//Sets width and height for the textbox
textbox.TextBoxFormat.Width = 150;
textbox.TextBoxFormat.Height = 75;
//Adds new text to the textbox body
IWParagraph textboxParagraph = textbox.TextBoxBody.AddParagraph();
//Adds new text to the textbox paragraph
textboxParagraph.AppendText("Text inside text box");
//Sets wrapping style for the textbox
textbox.TextBoxFormat.TextWrappingStyle = TextWrappingStyle.Behind;
//Sets horizontal and vertical position for the textbox
textbox.TextBoxFormat.HorizontalPosition = 200;
textbox.TextBoxFormat.VerticalPosition = 200;
//Sets horizontal and vertical origin for the textbox
textbox.TextBoxFormat.VerticalOrigin = VerticalOrigin.Page;
textbox.TextBoxFormat.HorizontalOrigin = HorizontalOrigin.Page;
//Adds specified textbox to the nested group shape
nestedGroupShape.Add(textbox);
//Appends new shape to the document
shape = new Shape(document, AutoShapeType.Oval);
//Sets height and width for the new shape
shape.Height = 100;
shape.Width = 150;
//Sets horizontal and vertical position for the shape
shape.HorizontalPosition = 200;
shape.VerticalPosition = 72;
//Sets horizontal and vertical origin for the shape
shape.HorizontalOrigin = HorizontalOrigin.Page;
shape.VerticalOrigin = VerticalOrigin.Page;
//Sets horizontal and vertical position for the nested group shape
nestedGroupShape.HorizontalPosition = 72;
nestedGroupShape.VerticalPosition = 72;

```



```
//Adds specified shape to the nested group shape
nestedGroupShape.Add(shape);
//Adds nested group shape to the group shape of the paragraph
groupShape.Add(nestedGroupShape);
groupShape.HorizontalPosition = 142;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Ungrouping shapes

You can ungroup the group shapes in the Word document to preserve each shape as individual item.

The following code example illustrates how to ungroup the group shape in Word document.

C#

```
//Loads the template document
WordDocument document = new WordDocument("Template.docx");
//Gets the last paragraph
WParagraph lastParagraph = document.LastParagraph;
//Iterates through the paragraph items to get the group shape
for (int i = 0; i < lastParagraph.ChildEntities.Count; i++)
{
    if (lastParagraph.ChildEntities[i] is GroupShape)
    {
        GroupShape groupShape = lastParagraph.ChildEntities[i] as GroupShape;
        //Ungroup the child shapes in the group shape
        groupShape.Ungroup();
        break;
    }
}
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Loads the template document
Dim document As WordDocument = New WordDocument("Template.docx")
'Gets the last paragraph
Dim lastParagraph As WParagraph = document.LastParagraph
'Iterates through the paragraph items to get the group shape
For i As Integer = 0 To lastParagraph.ChildEntities.Count - 1
    If TypeOf lastParagraph.ChildEntities(i) Is GroupShape Then
        Dim groupShape As GroupShape = TryCast(lastParagraph.ChildEntities(i),
        GroupShape)
```

```

groupShape.Ungroup()
Exit For
End If
Next
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Template.docx");
//Loads the template document
WordDocument document = new WordDocument(fileStream);
//Gets the last paragraph
WParagraph lastParagraph = document.LastParagraph;
//Iterates through the paragraph items to get the group shape
for (int i = 0; i < lastParagraph.ChildEntities.Count; i++)
{
    if (lastParagraph.ChildEntities[i] is GroupShape)
    {
        GroupShape groupShape = lastParagraph.ChildEntities[i] as GroupShape;
        //Ungroup the child shapes in the group shape
        groupShape.Ungroup();
        break;
    }
}
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "GroupShape.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

FileStream fileStream = new FileStream(@"Template.docx", FileMode.Open,
    FileAccess.ReadWrite);
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatTypeAutomatic);
//Gets the last paragraph
WParagraph lastParagraph = document.LastParagraph;
//Iterates through the paragraph items to get the group shape
for (int i = 0; i < lastParagraph.ChildEntities.Count; i++)
{
    if (lastParagraph.ChildEntities[i] is GroupShape)
    {
        GroupShape groupShape = lastParagraph.ChildEntities[i] as GroupShape;
        //Ungroup the child shapes in the group shape
    }
}

```

```

groupShape.Ungroup();
break;
}
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

//App is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("XamarinFormsApp.Assets.Template.docx");
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatTypeAutomatic);
//Gets the last paragraph
WParagraph lastParagraph = document.LastParagraph;
//Iterates through the paragraph items to get the group shape
for (int i = 0; i < lastParagraph.ChildEntities.Count; i++)
{
    if (lastParagraph.ChildEntities[i] is GroupShape)
    {
        GroupShape groupShape = lastParagraph.ChildEntities[i] as GroupShape;
        //Ungroup the child shapes in the group shape
        groupShape.Ungroup();
        break;
    }
}
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Working with Mail merge

Mail merge is a process of merging data from data source to a Word template document. The **WMergeField** class provides support to bind template document and data source. The **WMergeField** instance is replaced with the actual data retrieved from data source for the given merge field name in a template document.

The following data sources are supported by Essential DocIO for performing Mail merge:

- String Arrays
- ADO.NET objects
- Dynamic objects
- .NET objects

Mail merge process

The mail merge process involves three documents:

1. **Template Word document:** This document contains the static or templated text and graphics along with the merge fields (that are placeholders) for replacing dynamic data.
2. **Data source:** This represents file or database containing data to replace the merge fields in template Word document.
3. **Final merged document:** This resultant document is a combination of the template Word document and the data from data source.

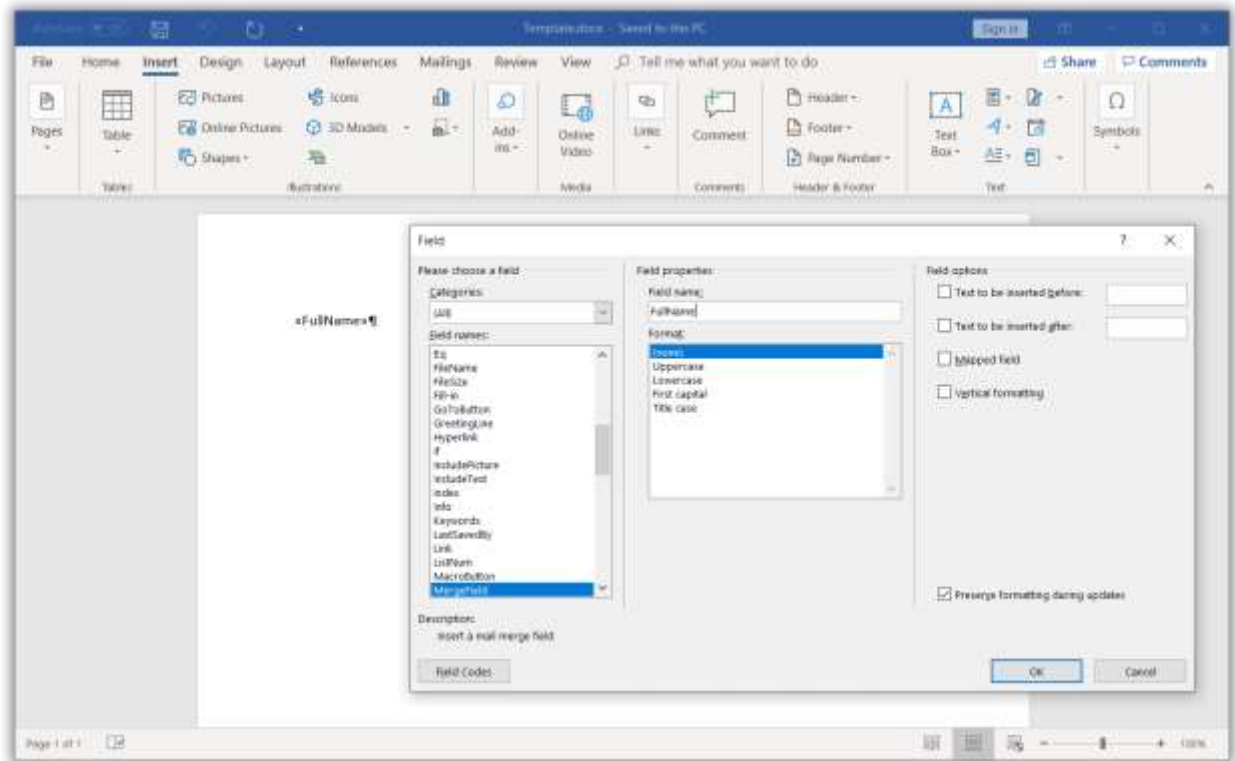
Tips: 1. You can use conditional fields ([IF](#), [Formula](#)) combined with merge fields, when you require intelligent decisions in addition to simple mail merge (replace merge fields with result text). To use conditional fields, execute mail merge and then update fields in the Word document using `UpdateDocumentFields` API.

Tips: 2. You can replace the fields ([IF](#), [Formula](#)) combined with merge fields, with its most recent result and **generates the plain Word document** by unlinking the fields. Refer to this [link](#) for more information.

Create Word document template

You can create a template document with merge fields by using any Word editor application, like Microsoft Word. By using Word editor application, you can take the advantage of the visual interface to design unique layout, formatting, and more for your Word document template interactively.

The following screenshot shows how to insert a merge field in the Word document by **using the Microsoft Word**.



You need to add a prefix ("Image:") to the merge field name for merging an image in the place of a merge field.

For example: The merge field name should be like "Image:Photo" (<<Image:MergeFieldName>>)

You can **create Word document template programmatically** by adding merge fields to the Word document using Essential DocIO.

The following code example shows how to create a merge field in the Word document.

C#

```
//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds a section and a paragraph in the document
document.EnsureMinimal();
//Appends merge field to the last paragraph.
document.LastParagraph.AppendField("FullName", FieldType.FieldMergeField);
//Saves and closes the WordDocument instance.
document.Save("Template.docx");
document.Close();
```

VB.NET

```
'Creates an instance of a WordDocument
Dim document As WordDocument = New WordDocument
'Adds a section and a paragraph in the document
document.EnsureMinimal()
'Appends merge field to the last paragraph.
document.LastParagraph.AppendField("FullName", FieldType.FieldMergeField)
'Saves and closes the WordDocument instance.
```

```
document.Save("Template.docx")
document.Close()
```

UWP

```
//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds a section and a paragraph in the document
document.EnsureMinimal();
//Appends merge field to the last paragraph.
document.LastParagraph.AppendField("FullName", FieldType.FieldMergeField);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Template.docx");
document.Close();
//Refer to the following link to save Word document in UWP platform.
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds a section and a paragraph in the document
document.EnsureMinimal();
//Appends merge field to the last paragraph.
document.LastParagraph.AppendField("FullName", FieldType.FieldMergeField);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the Word document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Template.docx");
```

XAMARIN

```
//Creates an instance of a WordDocument
WordDocument document = new WordDocument();
//Adds a section and a paragraph in the document
document.EnsureMinimal();
//Appends merge field to the last paragraph.
document.LastParagraph.AppendField("FullName", FieldType.FieldMergeField);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the Word document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Template.docx",
"application/msword", stream);
```

```
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform.
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Execute mail merge

The following code example shows how to perform mail merge in above Word document template using string arrays as data source.

C#

```
//Opens the template document
WordDocument document = new WordDocument("Template.docx");
string[] fieldNames = new string[] { "FullName" };
string[] fieldValues = new string[] { "Nancy Davolio" };
//Performs the mail merge
document.MailMerge.Execute(fieldNames, fieldValues);
//Saves and closes the WordDocument instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Opens the template document
Dim document As New WordDocument("Template.docx")
Dim fieldNames As String() = New String() {"FullName"}
Dim fieldValues As String() = New String() {"Nancy Davolio"}
'Performs the mail merge
document.MailMerge.Execute(fieldNames, fieldValues)
'Saves and closes the WordDocument instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates an instance of a WordDocument
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new WordDocument();
document.Open(assembly.GetManifestResourceStream("Sample.Assets.Template.doc
x"), FormatType.Docx);
string[] fieldNames = new string[] { "FullName" };
string[] fieldValues = new string[] { "Nancy Davolio" };
//Performs the mail merge
document.MailMerge.Execute(fieldNames, fieldValues);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
document.Close();
//Refer to the following link to save Word document in UWP platform.
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

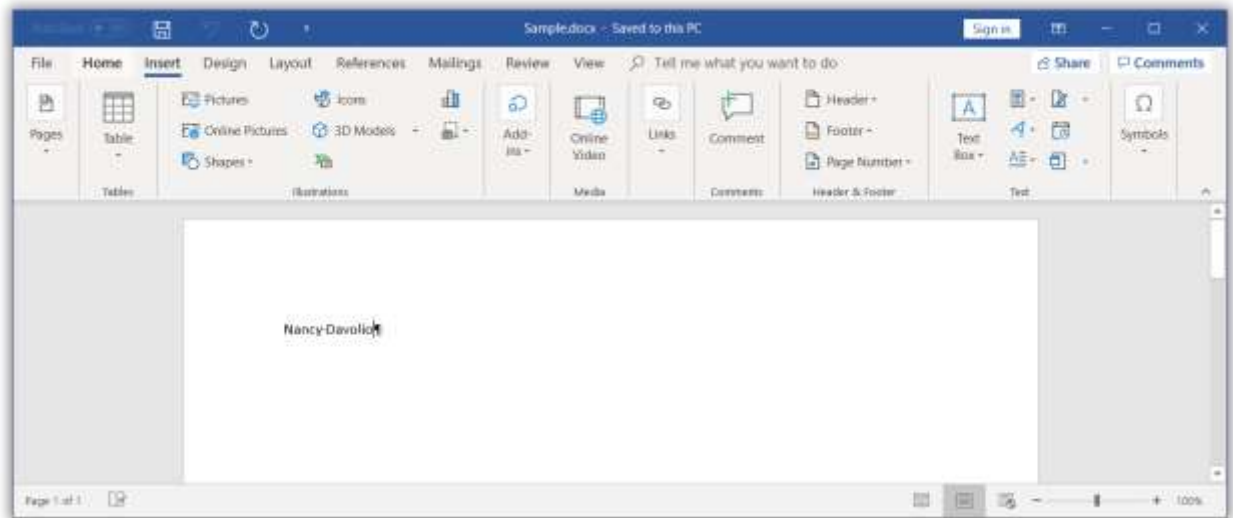
ASP.NET CORE

```
//Opens the template document
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
string[] fieldNames = new string[] { "FullName" };
string[] fieldValues = new string[] { "Nancy Davolio" };
//Performs the mail merge
document.MailMerge.Execute(fieldNames, fieldValues);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the Word document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Opens the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx"),
FormatType.Docx);
string[] fieldNames = new string[] { "FullName" };
string[] fieldValues = new string[] { "Nancy Davolio" };
//Performs the mail merge
document.MailMerge.Execute(fieldNames, fieldValues);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Closes the document
document.Close();
//Download the helper files from the following link to save the stream as
file and open the file for viewing in Xamarin platform.
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

By executing the previous code example, it generates the resultant Word document as follows.



Simple Mail merge

The `MailMerge` class provides various overloads for the `Execute` method to perform Mail merge from various data sources. For further information, click [here](#).

Performing Mail merge for a group

You can perform Mail merge and append multiple records from data source within a specified region to a template document. For further information, click [here](#).

Performing Nested Mail merge for group

You can perform nested Mail merge with relational or hierarchical data source and independent data tables in a template document. For further information, click [here](#).

Performing Mail merge with dynamic objects

Essential DocIO allows you to perform Mail merge with the dynamic objects. For further information, click [here](#).

Performing Mail merge with business objects

You can perform Mail merge with business objects in a template document. For further information, click [here](#).

Performing Nested Mail merge with relational data objects

Essential DocIO supports performing nested Mail merge with implicit relational data objects without any explicit relational commands by using the `ExecuteNestedGroup` overload method. For further information, click [here](#).

Event support for mail merge

The `MailMerge` class provides event support to customize the document contents and merging image data during the Mail merge process. The following events are supported by Essential DocIO in Mail merge process:

- `MergeField`: Occurs when a **Mail merge field** except image Mail merge field is encountered.
- `MergeImageField`: Occurs when an **image Mail merge field** is encountered.
- `BeforeClearField`: Occurs when an **unmerged field** is encountered.
- `BeforeClearGroupField`: Occurs when an **unmerged group field** is encountered.

MergeField event

You can customize the merging text during Mail merge process by using the `MergeField` event. For further information, click [here](#).

MergeImageField event

You can customize the merging image during Mail merge process by using the `MergeImageField` event. For further information, click [here](#).

BeforeClearField event

You can get the unmerged fields during Mail merge process by using the `BeforeClearField` event. For further information, click [here](#).

BeforeClearGroupField event

You can get the unmerged groups during Mail merge process by using the `BeforeClearGroupField` event. For further information, click [here](#).

Mail merge options

The `MailMerge` class allows you to customize the Mail merge process with the following options:

Field mapping

You can automatically map the merge field names with data source column names during Mail merge process. For further information, click [here](#).

Retrieving the merge field names

You can retrieve the merge field names and also merge field group names in the Word document. For further information, click [here](#).

Removing empty paragraphs

You can remove the empty paragraphs when the paragraph has a merge field item without any data during Mail merge process. For further information, click [here](#).

Removing empty merge fields

You can remove or keep the unmerged merge fields in the output document based on the `ClearFields` property on each mail merge execution. For further information, click [here](#).

Restart numbering in lists

You can restart the list numbering in a Word document during Mail merge. For further information, click [here](#).

Find and Replace

You can search a particular text you like to change and replace it with another text or part of the document.

Finding contents in a Word document

You can find the first occurrence of a particular text within a single paragraph in the document by using `Find` method and its next occurrence by using `FindNext` method. You can also find a particular text pattern in the document.

The following code example illustrates how to find a particular text and its next occurrence in the document.

C#

```
//Loads the template document  
WordDocument document = new WordDocument("Template.docx", FormatType.Docx);
```

```

//Finds the first occurrence of a particular text in the document
TextSelection textSelection = document.Find("as graphical contents", false,
true);
//Gets the found text as single text range
WTextRange textRange = textSelection.GetAsOneRange();
//Modifies the text
textRange.Text = "Replaced text";
//Sets highlight color
textRange.CharacterFormat.HighlightColor = Color.Yellow;
//Finds the next occurrence of a particular text from the previous paragraph
textSelection = document.FindNext(textRange.OwnerParagraph, "paragraph",
true, false);
//Gets the found text as single text range
WTextRange range = textSelection.GetAsOneRange();
//Sets bold formatting
range.CharacterFormat.Bold = true;
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx", FormatType.Docx)
'Find the first occurrence of a particular text in the document
Dim textSelection As TextSelection = document.Find("as graphical contents",
False, True)
'Gets the found text as single text range
Dim textRange As WTextRange = textSelection.GetAsOneRange()
'Modifies the text
textRange.Text = "Replaced text"
'Sets highlight color
textRange.CharacterFormat.HighlightColor = Color.Yellow
'Finds the next occurrence of a particular text from the previous paragraph
textSelection = document.FindNext(textRange.OwnerParagraph, "paragraph",
True, False)
'Gets the found text as single text range
Dim range As WTextRange = textSelection.GetAsOneRange()
'Sets bold formatting
range.CharacterFormat.Bold = True
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx
"), FormatType.Docx);
//Finds the first occurrence of a particular text in the document
TextSelection textSelection = document.Find("as graphical contents", false,
true);
//Gets the found text as single text range
WTextRange textRange = textSelection.GetAsOneRange();

```

```

//Modifies the text
textRange.Text = "Replaced text";
//Sets highlight color
textRange.CharacterFormat.HighlightColor = Color.Yellow;
//Finds the next occurrence of a particular text from the previous paragraph
textSelection = document.FindNext(textRange.OwnerParagraph, "paragraph",
true, false);
//Gets the found text as single text range
WTextRange range = textSelection.GetAsOneRange();
//Sets bold formatting
range.CharacterFormat.Bold = true;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads the template document
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Finds the first occurrence of a particular text in the document
TextSelection textSelection = document.Find("as graphical contents", false,
true);
//Gets the found text as single text range
WTextRange textRange = textSelection.GetAsOneRange();
//Modifies the text
textRange.Text = "Replaced text";
//Sets highlight color
textRange.CharacterFormat.HighlightColor = Color.Yellow;
//Finds the next occurrence of a particular text from the previous paragraph
textSelection = document.FindNext(textRange.OwnerParagraph, "paragraph",
true, false);
//Gets the found text as single text range
WTextRange range = textSelection.GetAsOneRange();
//Sets bold formatting
range.CharacterFormat.Bold = true;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Loads the template document

```

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Templat
e.docx"), FormatType.Docx);
//Finds the first occurrence of a particular text in the document
TextSelection textSelection = document.Find("as graphical contents", false,
true);
//Gets the found text as single text range
WTextRange textRange = textSelection.GetAsOneRange();
//Modifies the text
textRange.Text = "Replaced text";
//Sets highlight color
textRange.CharacterFormat.HighlightColor = Color.Yellow;
//Finds the next occurrence of a particular text from the previous paragraph
textSelection = document.FindNext(textRange.OwnerParagraph, "paragraph",
true, false);
//Gets the found text as single text range
WTextRange range = textSelection.GetAsOneRange();
//Sets bold formatting
range.CharacterFormat.Bold = true;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

You can find all the occurrence of a particular text within a single paragraph in the document by using `FindAll` method.

The following code example illustrates how to find all the occurrences of a particular text in the document.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Template.docx", FormatType.Docx);
//Finds all the occurrences of a particular text
TextSelection[] textSelections = document.FindAll("paragraph", false, true);
foreach (TextSelection textSelection in textSelections)
{
//Gets the found text as single text range and sets highlight color
WTextRange textRange = textSelection.GetAsOneRange();
textRange.CharacterFormat.HighlightColor = Color.YellowGreen;
}
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx", FormatType.Docx)
'Finds all the occurrences of a particular text
Dim textSelections As TextSelection() = document.FindAll("paragraph", False,
True)
For Each textSelection As TextSelection In textSelections
'Gets the found text as single text range and sets highlight color
Dim textRange As WTextRange = textSelection.GetAsOneRange()
textRange.CharacterFormat.HighlightColor = Color.YellowGreen
Next
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx
"), FormatType.Docx);
//Finds all the occurrences of a particular text
TextSelection[] textSelections = document.FindAll("paragraph", false, true);
foreach (TextSelection textSelection in textSelections)
{
//Gets the found text as single text range and sets highlight color
WTextRange textRange = textSelection.GetAsOneRange();
textRange.CharacterFormat.HighlightColor = Color.YellowGreen;
}
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads the template document
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Finds all the occurrences of a particular text
TextSelection[] textSelections = document.FindAll("paragraph", false, true);
foreach (TextSelection textSelection in textSelections)
{
//Gets the found text as single text range and sets highlight color
WTextRange textRange = textSelection.GetAsOneRange();
textRange.CharacterFormat.HighlightColor = Color.YellowGreen;
}

```

```
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Templat
e.docx"), FormatType.Docx);
//Finds all the occurrences of a particular text
TextSelection[] textSelections = document.FindAll("paragraph", false, true);
foreach (TextSelection textSelection in textSelections)
{
//Gets the found text as single text range and sets highlight color
WTextRange textRange = textSelection.GetAsOneRange();
textRange.CharacterFormat.HighlightColor = Color.YellowGreen;
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

You can find the first occurrence of a particular text extended to several paragraphs in the document by using `FindSingleLine` method and its next occurrence by using `FindNextSingleLine` method.

The following code example illustrates how to find a particular text extended to several paragraphs in the Word document.

C#

```
//Loads the template document
WordDocument document = new WordDocument("Template.docx", FormatType.Docx);
//Finds the first occurrence of a particular text extended to several
paragraphs in the document
TextSelection[] textSelections = document.FindSingleLine("First paragraph
Second paragraph", true, false);
WParagraph paragraph = null;
foreach (TextSelection textSelection in textSelections)
{
//Gets the found text as single text range and set highlight color
WTextRange textRange = textSelection.GetAsOneRange();
```

```

textRange.CharacterFormat.HighlightColor = Color.YellowGreen;
paragraph = textRange.OwnerParagraph;
}
//Finds the next occurrence of a particular text extended to several paragraphs in the document
textSelections = document.FindNextSingleLine(paragraph, "First paragraph Second paragraph", true, false);
foreach (TextSelection textSelection in textSelections)
{
//Gets the found text as single text range and sets italic formatting
WTextRange text = textSelection.GetAsOneRange();
text.CharacterFormat.Italic = true;
}
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx", FormatType.Docx)
'Finds the first occurrence of a particular text extended to several paragraphs in the document
Dim textSelections As TextSelection() = document.FindSingleLine("First paragraph Second paragraph", True, False)
Dim paragraph As WParagraph = Nothing
For Each textSelection As TextSelection In textSelections
'Gets the found text as single text range and sets highlight color
Dim textRange As WTextRange = textSelection.GetAsOneRange()
textRange.CharacterFormat.HighlightColor = Color.YellowGreen
paragraph = textRange.OwnerParagraph
Next
'Finds the next occurrence of a particular text extended to several paragraphs in the document
textSelections = document.FindNextSingleLine(paragraph, "First paragraph Second paragraph", True, False)
For Each textSelection As TextSelection In textSelections
'Gets the found text as single text range and sets italic formatting
Dim text As WTextRange = textSelection.GetAsOneRange()
text.CharacterFormat.Italic = True
Next
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx"), FormatType.Docx);
//Finds the first occurrence of a particular text extended to several paragraphs in the document
TextSelection[] textSelections = document.FindSingleLine("First paragraph Second paragraph", true, false);

```



```

WParagraph paragraph = null;
foreach (TextSelection textSelection in textSelections)
{
    //Gets the found text as single text range and set highlight color
    WTextRange textRange = textSelection.GetAsOneRange();
    textRange.CharacterFormat.HighlightColor = Color.YellowGreen;
    paragraph = textRange.OwnerParagraph;
}
//Finds the next occurrence of a particular text extended to several
paragraphs in the document
textSelections = document.FindNextSingleLine(paragraph, "First paragraph
Second paragraph", true, false);
foreach (TextSelection textSelection in textSelections)
{
    //Gets the found text as single text range and sets italic formatting
    WTextRange text = textSelection.GetAsOneRange();
    text.CharacterFormat.Italic = true;
}
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads the template document
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Finds the first occurrence of a particular text extended to several
paragraphs in the document
TextSelection[] textSelections = document.FindSingleLine("First paragraph
Second paragraph", true, false);
WParagraph paragraph = null;
foreach (TextSelection textSelection in textSelections)
{
    //Gets the found text as single text range and set highlight color
    WTextRange textRange = textSelection.GetAsOneRange();
    textRange.CharacterFormat.HighlightColor = Color.YellowGreen;
    paragraph = textRange.OwnerParagraph;
}
//Finds the next occurrence of a particular text extended to several
paragraphs in the document
textSelections = document.FindNextSingleLine(paragraph, "First paragraph
Second paragraph", true, false);
foreach (TextSelection textSelection in textSelections)
{
    //Gets the found text as single text range and sets italic formatting
    WTextRange text = textSelection.GetAsOneRange();
    text.CharacterFormat.Italic = true;
}

```

```

}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Templat
e.docx"), FormatType.Docx);
//Finds the first occurrence of a particular text extended to several
paragraphs in the document
TextSelection[] textSelections = document.FindSingleLine("First paragraph
Second paragraph", true, false);
WParagraph paragraph = null;
foreach (TextSelection textSelection in textSelections)
{
//Gets the found text as single text range and set highlight color
WTextRange textRange = textSelection.GetAsOneRange();
textRange.CharacterFormat.HighlightColor = Color.YellowGreen;
paragraph = textRange.OwnerParagraph;
}
//Finds the next occurrence of a particular text extended to several
paragraphs in the document
textSelections = document.FindNextSingleLine(paragraph, "First paragraph
Second paragraph", true, false);
foreach (TextSelection textSelection in textSelections)
{
//Gets the found text as single text range and sets italic formatting
WTextRange text = textSelection.GetAsOneRange();
text.CharacterFormat.Italic = true;
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Replacing the Search results

You can replace a particular text with another text, part of a document or entire document by using **Replace** method.

The following code example illustrates how to replace a particular text.

C#

```
//Loads a template document
WordDocument document = new WordDocument("Template.docx", FormatType.Docx);
//Finds the first occurrence of a particular text in the document
TextSelection selection = document.Find("contents", false, false);
//Initializes text body part
TextBodyPart bodyPart = new TextBodyPart(selection);
//Replaces a particular text with the text body part
document.Replace("paragraph", bodyPart, false, true, true);
//Saves and closes the document
document.Save("Replace.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Loads a template document
Dim document As New WordDocument("Template.docx", FormatType.Docx)
'Finds the first occurrence of a particular text in the document
Dim selection As TextSelection = document.Find("contents", False, False)
'Initializes text body part
Dim bodyPart As New TextBodyPart(selection)
'Replaces a particular text with the text body part
document.Replace("paragraph", bodyPart, False, True, True)
'Saves and closes the document
document.Save("Replace.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Loads a template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx"),
FormatType.Docx);
//Finds the first occurrence of a particular text in the document
TextSelection selection = document.Find("contents", false, false);
//Initializes text body part
TextBodyPart bodyPart = new TextBodyPart(selection);
//Replaces a particular text with the text body part
document.Replace("paragraph", bodyPart, false, true, true);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Replace.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Loads a template document
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open,
    FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Finds the first occurrence of a particular text in the document
TextSelection selection = document.Find("contents", false, false);
//Initializes text body part
TextBodyPart bodyPart = new TextBodyPart(selection);
//Replaces a particular text with the text body part
document.Replace("paragraph", bodyPart, false, true, true);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Replace.docx");
```

XAMARIN

```
//Loads a template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Templat
e.docx"), FormatType.Docx);
//Finds the first occurrence of a particular text in the document
TextSelection selection = document.Find("contents", false, false);
//Initializes text body part
TextBodyPart bodyPart = new TextBodyPart(selection);
//Replaces a particular text with the text body part
document.Replace("paragraph", bodyPart, false, true, true);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Replace.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

You can specify to replace only the first occurrence of the specified text by setting `ReplaceFirst` property of `WordDocument` class to true.

The following code example illustrates how to replace the first occurrence of a particular text.

C#

```
//Loads a template document
WordDocument document = new WordDocument("Template.docx", FormatType.Docx);
//Sets to replace only the first occurrence of a particular text
document.ReplaceFirst = true;
```

```
//Finds the first occurrence of a particular text in the document
TextSelection selection = document.Find("contents", false, false);
//Initializes text body part
TextBodyPart bodyPart = new TextBodyPart(selection);
//Replaces a particular text with the text body part
document.Replace("paragraph", bodyPart, false, true, true);
//Saves and closes the document
document.Save("Replace.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Loads a template document
Dim document As New WordDocument("Template.docx", FormatType.Docx)
'Sets to replace only the first occurrence of a particular text
document.ReplaceFirst = True
'Finds the first occurrence of a particular text in the document
Dim selection As TextSelection = document.Find("contents", False, False)
'Initializes text body part
Dim bodyPart As New TextBodyPart(selection)
'Replaces a particular text with the text body part
document.Replace("paragraph", bodyPart, False, True, True)
'Saves and closes the document
document.Save("Replace.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Loads a template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx"),
FormatType.Docx);
//Sets to replace only the first occurrence of a particular text
document.ReplaceFirst = true;
//Finds the first occurrence of a particular text in the document
TextSelection selection = document.Find("contents", false, false);
//Initializes text body part
TextBodyPart bodyPart = new TextBodyPart(selection);
//Replaces a particular text with the text body part
document.Replace("paragraph", bodyPart, false, true, true);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Replace.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Loads a template document
```

```

FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open,
    FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Sets to replace only the first occurrence of a particular text
document.ReplaceFirst = true;
//Finds the first occurrence of a particular text in the document
TextSelection selection = document.Find("contents", false, false);
//Initializes text body part
TextBodyPart bodyPart = new TextBodyPart(selection);
//Replaces a particular text with the text body part
document.Replace("paragraph", bodyPart, false, true, true);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Replace.docx");

```

XAMARIN

```

//Loads a template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Templat
e.docx"), FormatType.Docx);
//Sets to replace only the first occurrence of a particular text
document.ReplaceFirst = true;
//Finds the first occurrence of a particular text in the document
TextSelection selection = document.Find("contents", false, false);
//Initializes text body part
TextBodyPart bodyPart = new TextBodyPart(selection);
//Replaces a particular text with the text body part
document.Replace("paragraph", bodyPart, false, true, true);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Replace.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

The following code example illustrates how to replace a particular text with a Word document.

C#

```

//Loads a template document
WordDocument document = new WordDocument("SourceTemplate.docx",
FormatType.Docx);
//Gets the document to replace the text

```

```
IWordDocument replaceDocument = new WordDocument("Template.docx");
//Replaces a particular text with another document
document.Replace("paragraph", replaceDocument, false, true, true);
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Loads a template document
Dim document As New WordDocument("SourceTemplate.docx", FormatType.Docx)
'Gets the document to replace the text
Dim replaceDocument As IWordDocument = New WordDocument("Template.docx")
'Replaces a particular text with another document
document.Replace("paragraph", replaceDocument, False, True, True)
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Loads a template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.SourceTemplat
e.docx"), FormatType.Docx);
//Gets the document to replace the text
IWordDocument replaceDocument = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx
"), FormatType.Docx);
//Replaces a particular text with another document
document.Replace("paragraph", replaceDocument, false, true, true);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Loads a template document
FileStream fileStreamPath1 = new FileStream("SourceTemplate.docx",
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath1, FormatType.Docx);
//Gets the document to replace the text
FileStream fileStreamPath2 = new FileStream("Template.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
IWordDocument replaceDocument = new WordDocument(fileStreamPath2,
FormatType.Docx);
//Replaces a particular text with another document
```

```
document.Replace("paragraph", replaceDocument, false, true, true);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Loads a template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.SourceT
emplate.docx"), FormatType.Docx);
//Gets the document to replace the text
IWordDocument replaceDocument = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Templat
e.docx"), FormatType.Docx);
//Replaces a particular text with another document
document.Replace("paragraph", replaceDocument, false, true, true);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

You can replace a particular text extended to several paragraphs in a document with another text or part of a document by using `ReplaceSingleLine` method.

The following code example illustrates how to replace the text extended to several paragraphs with simple text.

C#

```
//Loads a template document
WordDocument document = new WordDocument("Template.docx", FormatType.Docx);
//Replaces the text extended to two paragraphs with simple text
document.ReplaceSingleLine("First paragraph Second paragraph", "Replaced
paragraph", true, false);
//Saves and closes the document
document.Save("Replace.docx", FormatType.Docx);
document.Close();
```

VB.NET


```

'Loads a template document
Dim document As New WordDocument("Template.docx", FormatType.Docx)
'Replaces the text extended to two paragraphs with simple text
document.ReplaceSingleLine("First paragraph Second paragraph", "Replaced
paragraph", True, False)
'Saves and closes the document
document.Save("Replace.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Loads a template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx
"), FormatType.Docx);
//Replaces the text extended to two paragraphs with simple text
document.ReplaceSingleLine("First paragraph Second paragraph", "Replaced
paragraph", true, false);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Replace.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads a template document
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Replaces the text extended to two paragraphs with simple text
document.ReplaceSingleLine("First paragraph Second paragraph", "Replaced
paragraph", true, false);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Replace.docx");

```

XAMARIN

```

//Loads a template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Templat
e.docx"), FormatType.Docx);

```

```
//Replaces the text extended to two paragraphs with simple text
document.ReplaceSingleLine("First paragraph Second paragraph", "Replaced
paragraph", true, false);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Replace.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Working with Table Of Contents

[Table of contents](#) (TOC) is used to provide an outline of the Word document. By default table of contents will be created automatically from heading styles.

You can add the TOC into the paragraph by specifying the `LowerHeadingLevel` and `UpperHeadingLevel`. The heading level range must be from 1 to 9.

Basically TOC determines the TOC entries based on the TOC switches.

S.No	Switches	Description
1	\o	Builds a table of contents from paragraphs formatted with built-in heading styles.
2	\h	Inserts TOC entries as hyperlinks.
3	\n	Omits page numbers from the table of contents.
4	\p	Specifies the characters that separate a TOC entry and its page number. The default is a tab with leader dots.
5	\t	Builds a table of contents from paragraphs formatted with specified styles other than the built-in heading styles

Adding a TOC field

The following code example shows how to add a table of contents (TOC) in Word document.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds the section into the Word document
IWSection section = document.AddSection();
string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
```

```

//Adds the paragraph into the created section
IWParagraph paragraph = section.AddParagraph();
//Appends the TOC field with LowerHeadingLevel and UpperHeadingLevel to
determines the TOC entries
paragraph.AppendTOC(1, 3);
//Adds the section into the Word document
section = document.AddSection();
//Adds the paragraph into the created section
paragraph = section.AddParagraph();
//Adds the text for the headings
paragraph.AppendText("First Chapter");
//Sets a built-in heading style.
paragraph.ApplyStyle(BuiltinStyle.Heading1);
//Adds the text into the paragraph
section.AddParagraph().AppendText(paraText);
//Adds the section into the Word document
section = document.AddSection();
//Adds the paragraph into the created section
paragraph = section.AddParagraph();
//Adds the text for the headings
paragraph.AppendText("Second Chapter");
//Sets a built-in heading style.
paragraph.ApplyStyle(BuiltinStyle.Heading2);
//Adds the text into the paragraph
section.AddParagraph().AppendText(paraText);
//Adds the section into the Word document
section = document.AddSection();
//Adds the paragraph into the created section
paragraph = section.AddParagraph();
//Adds the text into the headings
paragraph.AppendText("Third Chapter");
//Sets a built-in heading style
paragraph.ApplyStyle(BuiltinStyle.Heading3);
//Adds the text into the paragraph.
section.AddParagraph().AppendText(paraText);
//Updates the table of contents
document.UpdateTableOfContents();
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds the section into the Word document
Dim section As IWSection = document.AddSection()
Dim paraText As String = "AdventureWorks Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company."
'Adds the paragraph into the created section
Dim paragraph As IWParagraph = section.AddParagraph()
'Appends the TOC field with LowerHeadingLevel and UpperHeadingLevel to
determines the TOC entries
paragraph.AppendTOC(1, 3)
'Adds the section into the Word document

```

```

section = document.AddSection()
'Adds the paragraph into the created section
paragraph = section.AddParagraph()
'Adds the text for the headings
paragraph.AppendText("First Chapter")
'Sets a built-in heading style
paragraph.ApplyStyle(BuiltinStyle.Heading1)
'Adds the text into the paragraph.
section.AddParagraph().AppendText(paraText)
'Adds the section into the Word document
section = document.AddSection()
'Adds the paragraph into the created section
paragraph = section.AddParagraph()
'Adds the text for the headings
paragraph.AppendText("Second Chapter")
'Sets a built-in heading style
paragraph.ApplyStyle(BuiltinStyle.Heading2)
'Adds the text into the paragraph
section.AddParagraph().AppendText(paraText)
'Adds the section into the Word document
section = document.AddSection()
'Adds the paragraph into the created section
paragraph = section.AddParagraph()
'Adds the text into the headings
paragraph.AppendText("Third Chapter")
'Sets a built-in heading style
paragraph.ApplyStyle(BuiltinStyle.Heading3)
'Adds the text into the paragraph
section.AddParagraph().AppendText(paraText)
'Updates the table of contents
document.UpdateTableOfContents()
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```
//DocIO supports Table of contents in WPF, Windows Forms platforms alone
```

ASP.NET CORE

```
//DocIO supports Table of contents in WPF, Windows Forms platforms alone
```

XAMARIN

```
//DocIO supports Table of contents in WPF, Windows Forms platforms alone
```

Updating table of contents

You can also update or re-build the TOC in an existing document or document created from the scratch.

Note: 1. Updating of TOC is not supported in Silverlight, WinRT, Universal, Xamarin, ASP.NET Core, Blazor and Windows Phone applications.

2. Updating TOC makes use of the Word to PDF layout engine that may lead to update incorrect page number due to its limitations.

The following code example shows how to update a TOC in an existing word document.

C#

```
//Opens an input word template
WordDocument document = new WordDocument(@"Template.docx");
//Updates the table of contents.
document.UpdateTableOfContents();
//Saves and closes the Word document instance.
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Opens an input word template
Dim document As New WordDocument("Template.docx")
'Updates the table of contents.
document.UpdateTableOfContents()
'Saves and closes the Word document instance.
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//DocIO supports Table of contents in WPF, Windows Forms platforms alone
```

ASP.NET CORE

```
//DocIO supports Table of contents in WPF, Windows Forms platforms alone
```

XAMARIN

```
//DocIO supports Table of contents in WPF, Windows Forms platforms alone
```

Creating table of contents with user-defined styles

The following code example shows how to create table of contents with user-defined styles instead of heading styles.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Creates a new custom styles
Style style = (WParagraphStyle)document.AddParagraphStyle("MyStyle");
style.CharacterFormat.Bold = true;
style.CharacterFormat.FontName = "Verdana";
style.CharacterFormat.FontSize = 25;
//Adds the section into the Word document
IWSection section = document.AddSection();
```

```

string paraText = "AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.";
//Adds the paragraph into the created section
IWParagraph paragraph = section.AddParagraph();
//Appends the TOC field with LowerHeadingLevel and UpperHeadingLevel to
determines the TOC entries
TableOfContent tableOfContents = paragraph.AppendTOC(1, 3);
tableOfContents.UseHeadingStyles = false;
//Sets the TOC level style based on the created TOC
tableOfContents.SetTOCLevelStyle(2, "MyStyle");
//Adds the section into the Word document
section = document.AddSection();
//Adds the paragraph into the created section
paragraph = section.AddParagraph();
//Adds the text for the headings
paragraph.AppendText("First Chapter");
//Sets the built-in heading style
paragraph.ApplyStyle("MyStyle");
//Adds the text into the paragraph
section.AddParagraph().AppendText(paraText);
//Adds the section into the Word document
section = document.AddSection();
//Adds the paragraph into the created section
paragraph = section.AddParagraph();
//Adds the text for the headings
paragraph.AppendText("Second Chapter");
//Sets the built-in heading style
paragraph.ApplyStyle(BuiltinStyle.Heading1);
//Adds the text to the paragraph
section.AddParagraph().AppendText(paraText);
//Adds the section into Word document
section = document.AddSection();
//Adds a paragraph to a created section
paragraph = section.AddParagraph();
//Adds the text for the headings
paragraph.AppendText("Third Chapter");
//Sets the built-in heading style
paragraph.ApplyStyle("MyStyle");
//Adds the text to the paragraph
section.AddParagraph().AppendText(paraText);
//Updates the table of contents
document.UpdateTableOfContents();
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Creates a new custom styles
Dim style As Style = DirectCast(document.AddParagraphStyle("MyStyle"),
WParagraphStyle)
style.CharacterFormat.Bold = True
style.CharacterFormat.FontName = "Verdana"

```

```

style.CharacterFormat.FontSize = 25
'Adds the section into the Word document
Dim section As IWSection = document.AddSection()
Dim paraText As String = "AdventureWorks Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company."
'Adds the paragraph into the created section
Dim paragraph As IWParagraph = section.AddParagraph()
'Appends the TOC field with LowerHeadingLevel and UpperHeadingLevel to
determine the TOC entries
Dim tableOfContents As TableOfContent = paragraph.AppendTOC(1, 3)
tableOfContents.UseHeadingStyles = False
'Sets the TOC level style based on the created TOC
tableOfContents.SetTOCLevelStyle(2, "MyStyle")
'Adds the section into the Word document
section = document.AddSection()
'Adds the paragraph into the created section
paragraph = section.AddParagraph()
'Adds the text for the headings
paragraph.AppendText("First Chapter")
'Sets the built-in heading style
paragraph.ApplyStyle("MyStyle")
'Adds the text into the paragraph
section.AddParagraph().AppendText(paraText)
'Adds the section into the Word document
section = document.AddSection()
'Adds the paragraph into the created section
paragraph = section.AddParagraph()
'Adds the text for the headings
paragraph.AppendText("Second Chapter")
'Sets the built-in heading style
paragraph.ApplyStyle(BuiltinStyle.Heading1)
'Adds the text to the paragraph
section.AddParagraph().AppendText(paraText)
'Adds the section into Word document
section = document.AddSection()
'Adds a paragraph to created section
paragraph = section.AddParagraph()
'Adds the text for the headings
paragraph.AppendText("Third Chapter")
'Sets the built-in heading style
paragraph.ApplyStyle("My style")
'Adds the text to the paragraph
section.AddParagraph().AppendText(paraText)
'Updates the table of contents
document.UpdateTableOfContents()
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```
//DocIO supports Table of contents in WPF, Windows Forms platforms alone
```

ASP.NET CORE

```
//DocIO supports Table of contents in WPF, Windows Forms platforms alone
```

XAMARIN

```
//DocIO supports Table of contents in WPF, Windows Forms platforms alone
```

Applying Watermark

Watermarks are text or pictures that appear behind the document text. You can access the watermark in the document by using the **Watermark** property of **WordDocument** class.

There are two types of watermarks: Text and Picture.

Text Watermark

You can add or modify text watermark in the Word document. **TextWatermark** class represents text watermark in the Word document.

The following code example illustrates how to add a text watermark to the Word document.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds a section and a paragraph in the document
document.EnsureMinimal();
IWParagraph paragraph = document.LastParagraph;
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Creates a new text watermark
TextWatermark textWatermark = new TextWatermark();
//Sets the created watermark to the document
document.Watermark = textWatermark;
//Sets the text watermark font size
textWatermark.Size = 72;
//Sets the text watermark layout to Horizontal
textWatermark.Layout = WatermarkLayout.Horizontal;
textWatermark.Semitransparent = false;
//Sets the text watermark text color
textWatermark.Color = Color.Black;
//Sets the text to text watermark text
textWatermark.Text = "TextWatermark";
document.Save("TextWatermark.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds a section and a paragraph in the document
document.EnsureMinimal()
Dim paragraph As IWParagraph = document.LastParagraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.")
'Creates a new text watermark
```



```

Dim textWatermark As New TextWatermark()
'Sets the text watermark to the document
document.Watermark = textWatermark
'Sets the text watermark font size
textWatermark.Size = 72
'Sets the text watermark layout to Horizontal
textWatermark.Layout = WatermarkLayout.Horizontal
textWatermark.Semitransparent = False
'Sets the text watermark text color
textWatermark.Color = Color.Black
'Sets the text to the text watermark
textWatermark.Text = "TextWatermark"
document.Save("TextWatermark.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds a section and a paragraph in the document
document.EnsureMinimal();
IWParagraph paragraph = document.LastParagraph;
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Creates a new text watermark
TextWatermark textWatermark = new TextWatermark();
//Sets the created watermark to the document
document.Watermark = textWatermark;
//Sets the text watermark font size
textWatermark.Size = 72;
//Sets the text watermark layout to Horizontal
textWatermark.Layout = WatermarkLayout.Horizontal;
textWatermark.Semitransparent = false;
//Sets the text watermark text color
textWatermark.Color = Color.Black;
//Sets the text to text watermark text
textWatermark.Text = "TextWatermark";
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "TextWatermark.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds a section and a paragraph in the document
document.EnsureMinimal();
IWParagraph paragraph = document.LastParagraph;

```

```

paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Creates a new text watermark
TextWatermark textWatermark = new TextWatermark("TextWatermark", "", 250,
100);
//Sets the created watermark to the document
document.Watermark = textWatermark;
//Sets the text watermark font size
textWatermark.Size = 72;
//Sets the text watermark layout to Horizontal
textWatermark.Layout = WatermarkLayout.Horizontal;
textWatermark.Semitransparent = false;
//Sets the text watermark text color
textWatermark.Color = Color.Black;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result_watermark1.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds a section and a paragraph in the document
document.EnsureMinimal();
IWParagraph paragraph = document.LastParagraph;
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Creates a new text watermark
TextWatermark textWatermark = new TextWatermark("TextWatermark", "", 250,
100);
//Sets the created watermark to the document
document.Watermark = textWatermark;
//Sets the text watermark font size
textWatermark.Size = 72;
//Sets the text watermark layout to Horizontal
textWatermark.Layout = WatermarkLayout.Horizontal;
textWatermark.Semitransparent = false;
//Sets the text watermark text color
textWatermark.Color = Syncfusion.Drawing.Color.Black;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TextWatermark.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform

```

[//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin](https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin)

Picture Watermark

You can add or modify picture watermark in the Word document. **PictureWatermark** class represents picture watermark in the Word document.

The following code example illustrates how to add a picture watermark to the Word document.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds a section and a paragraph in the document
document.EnsureMinimal();
IWParagraph paragraph = document.LastParagraph;
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Creates a new picture watermark
PictureWatermark picWatermark = new PictureWatermark();
//Sets the scaling to picture
picWatermark.Scaling = 120f;
picWatermark.Washout = true;
//Sets the picture watermark to document
document.Watermark = picWatermark;
//Sets the image to the picture watermark
picWatermark.Picture = Image.FromFile(ImagesPath + "Water lilies.jpg");
document.Save("PictureWatermark.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds a section and a paragraph in the document
document.EnsureMinimal()
Dim paragraph As IWParagraph = document.LastParagraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.")
'Creates a new picture watermark
Dim picWatermark As New PictureWatermark()
'Sets the scaling to picture
picWatermark.Scaling = 120.0F
picWatermark.Washout = True
Set the picture watermark to document
document.Watermark = picWatermark
Set the image to the picture watermark
picWatermark.Picture = Image.FromFile(ImagesPath + "Water lilies.jpg")
document.Save("PictureWatermark.docx", FormatType.Docx)
document.Close()
```

UWP

```
//DocIO supports picture watermark in Windows Forms, WPF, ASP.NET and
ASP.NET MVC platforms alone.
```

ASP.NET CORE

```
//DocIO supports picture watermark in Windows Forms, WPF, ASP.NET and
ASP.NET MVC platforms alone.
```

XAMARIN

```
//DocIO supports picture watermark in Windows Forms, WPF, ASP.NET and
ASP.NET MVC platforms alone.
```

Working with Comments

A comment is a note or annotation that an author or reviewer can add to a document. DocIO represents comment with **WComment** instance.

Note: The comment start and end ranges and dates can be preserved only on processing an existing document that already contains these information for each comment.

Adding a Comment

You can add a new comment to the Word document by using **AppendComment** method of **WParagraph** class.

The following code illustrates how to add a new comment to the document:

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds a section and a paragraph in the document
document.EnsureMinimal();
IWParagraph paragraph = document.LastParagraph;
//Appends text to the paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Adds comment to a paragraph
WComment comment = paragraph.AppendComment("comment test");
//Specifies the author of the comment
comment.Format.User = "Peter";
//Specifies the initial of the author
comment.Format.UserInitials = "St";
//Set the date and time for comment
comment.Format.DateTime = DateTime.Now;
//Saves and closes the Word document
document.Save("Comment.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds a section and a paragraph in the document
```

```

document.EnsureMinimal()
Dim paragraph As IParagraph = document.LastParagraph
'Appends text to the paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.")
'Adds comment to a paragraph
Dim comment As WComment = paragraph.AppendComment("comment test")
'Specifies the author of the comment
comment.Format.User = "Peter"
'Specifies the initial of the author
comment.Format.UserInitials = "St"
'Saves and closes the Word document
document.Save("Comment.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds a section and a paragraph in the document
document.EnsureMinimal();
IParagraph paragraph = document.LastParagraph;
//Appends text to the paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Adds comment to a paragraph
WComment comment = paragraph.AppendComment("comment test");
//Specifies the author of the comment
comment.Format.User = "Peter";
//Specifies the initial of the author
comment.Format.UserInitials = "St";
//Set the date and time for comment
comment.Format.DateTime = DateTime.Now;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Comment.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds a section and a paragraph in the document
document.EnsureMinimal();
IParagraph paragraph = document.LastParagraph;
//Appends text to the paragraph

```

```

paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Adds comment to a paragraph
WComment comment = paragraph.AppendComment("comment test");
//Specifies the author of the comment
comment.Format.User = "Peter";
//Specifies the initial of the author
comment.Format.UserInitials = "St";
//Set the date and time for comment
comment.Format.DateTime = DateTime.Now;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Comment.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds a section and a paragraph in the document
document.EnsureMinimal();
IWParagraph paragraph = document.LastParagraph;
//Appends text to the paragraph
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Adds comment to a paragraph
WComment comment = paragraph.AppendComment("comment test");
//Specifies the author of the comment
comment.Format.User = "Peter";
//Specifies the initial of the author
comment.Format.UserInitials = "St";
//Set the date and time for comment
comment.Format.DateTime = DateTime.Now;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Modifying a Comment

The following code illustrates how to modify the text of an existing comment in the Word document:

C#

```

WordDocument document = new WordDocument("Comment.docx");
//Iterates the comments in the Word document
foreach (WComment comment in document.Comments)
{
    //Modifies the last paragraph text of an existing comment when it is added
    by "Peter"
    if (comment.Format.User == "Peter")
        comment.TextBody.LastParagraph.Text = "Modified Comment Content";
}
document.Save("ModifiedComment.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

Dim document As New WordDocument("Comment.docx")
'Iterates the comments in the Word document
For Each comment As WComment In document.Comments
    'Modifies the last paragraph text of an existing comment when it is added by
    "Peter"
    If comment.Format.User = "Peter" Then
        comment.TextBody.LastParagraph.Text = "Modified Comment Content"
    End If
Next
document.Save("ModifiedComment.docx", FormatType.Docx)
document.Close()

```

UWP

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Comment.docx"
), FormatType.Docx);
//Iterates the comments in the Word document
foreach (WComment comment in document.Comments)
{
    //Modifies the last paragraph text of an existing comment when it is added
    by "Peter"
    if (comment.Format.User == "Peter")
        comment.TextBody.LastParagraph.Text = "Modified Comment Content";
}
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "ModifiedComment.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

FileStream fileStreamPath = new FileStream("Comment.docx", FileMode.Open,
    FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Iterates the comments in the Word document
foreach (WComment comment in document.Comments)
{
    //Modifies the last paragraph text of an existing comment when it is added
    by "Peter"
    if (comment.Format.User == "Peter")
        comment.TextBody.LastParagraph.Text = "Modified Comment Content";
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "ModifiedComment.docx");

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Comment
.docx"), FormatType.Docx);
//Iterates the comments in the Word document
foreach (WComment comment in document.Comments)
{
    //Modifies the last paragraph text of an existing comment when it is added
    by "Peter"
    if (comment.Format.User == "Peter")
        comment.TextBody.LastParagraph.Text = "Modified Comment Content";
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ModifiedComment.do
cx", "application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Removing Comments

You can either remove all the comments or a particular comment from the Word document.

The following code illustrates how to remove all the comments in Word document.

C#

```

WordDocument document = new WordDocument("Comment.docx");
//Removes all the comments in a Word document

```



```
document.Comments.Clear();
document.Save("Result.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
Dim document As New WordDocument("Comment.docx")
'Removes all the comments in a Word document
document.Comments.Clear()
document.Save("Result.docx", FormatType.Docx)
document.Close()
```

UWP

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Comment.docx"),
FormatType.Docx);
//Removes all the comments in a Word document
document.Comments.Clear();
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
FileStream filePath = new FileStream("Comment.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(filePath, FormatType.Docx);
//Removes all the comments in a Word document
document.Comments.Clear();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
```

XAMARIN

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Comment
.docx"), FormatType.Docx)
//Removes all the comments in a Word document
document.Comments.Clear();
```

```
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

The following code illustrates how to remove a particular comment from Word document.

C#

```
WordDocument document = new WordDocument("Comment.docx");
//Removes second comments from a document.
document.Comments.RemoveAt(1);
//Saves and closes the Word document
document.Save("Result.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
Dim document As New WordDocument("Comment.docx")
'Removes second comments from a document.
document.Comments.RemoveAt(1)
'Saves and closes the Word document
document.Save("Result.docx", FormatType.Docx)
document.Close()
```

UWP

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Comment.docx"
), FormatType.Docx);
//Removes second comments from a document.
document.Comments.RemoveAt(1);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```

FileStream filePath = new FileStream("Comment.docx", FileMode.Open,
    FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(filePath, FormatType.Docx);
//Removes second comments from a document.
document.Comments.RemoveAt(1);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Comment
.docx"), FormatType.Docx)
//Removes second comments from a document
document.Comments.RemoveAt(1);
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Working with Form Fields

You can create template document with form fields such as Text, Checkbox and Drop-Down. You can also open an existing template document and fill the form fields with the specified data.

The following are the types of form field in the Word document

- Checkbox – represented by an instance of WCheckBox
- Drop-down – represented by an instance of WDropDownFormField
- Text input – represented by an instance of WTextFormField

Check Box

You can add new Checkbox form field to a Word document by using `AppendCheckBox` method of `WParagraph` class.

The following code illustrates how to add new checkbox form field.

C#

```

//Creates a new Word document

```

```

WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendText("Gender\t");
//Appends new Checkbox
WCheckBox checkbox = paragraph.AppendCheckBox();
checkbox.Checked = false;
//Sets Checkbox size
checkbox.CheckBoxSize = 10;
checkbox.CalculateOnExit = true;
//Sets help text
checkbox.Help = "Help text";
paragraph.AppendText("Male\t");
checkbox = paragraph.AppendCheckBox();
checkbox.Checked = false;
checkbox.CheckBoxSize = 10;
checkbox.CalculateOnExit = true;
paragraph.AppendText("Female");
//Saves the Word document
document.Save("Checkbox.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As WParagraph = TryCast(section.AddParagraph(), WParagraph)
paragraph.AppendText("Gender" & vbTab)
'Appends new Checkbox
Dim checkbox As WCheckBox = paragraph.AppendCheckBox()
checkbox.Checked = False
'Sets Checkbox size
checkbox.CheckBoxSize = 10
checkbox.CalculateOnExit = True
'Sets help text
checkbox.Help = "Help text"
paragraph.AppendText("Male" & vbTab)
checkbox = paragraph.AppendCheckBox()
checkbox.Checked = False
checkbox.CheckBoxSize = 10
checkbox.CalculateOnExit = True
paragraph.AppendText("Female")
'Saves the Word document
document.Save("Checkbox.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document

```

```

WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendText("Gender\t");
//Appends new Checkbox
WCheckBox checkbox = paragraph.AppendCheckBox();
checkbox.Checked = false;
//Sets Checkbox size
checkbox.CheckBoxSize = 10;
checkbox.CalculateOnExit = true;
//Sets help text
checkbox.Help = "Help text";
paragraph.AppendText("Male\t");
checkbox = paragraph.AppendCheckBox();
checkbox.Checked = false;
checkbox.CheckBoxSize = 10;
checkbox.CalculateOnExit = true;
paragraph.AppendText("Female");
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Checkbox.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendText("Gender\t");
//Appends new Checkbox
WCheckBox checkbox = paragraph.AppendCheckBox();
checkbox.Checked = false;
//Sets Checkbox size
checkbox.CheckBoxSize = 10;
checkbox.CalculateOnExit = true;
//Sets help text
checkbox.Help = "Help text";
paragraph.AppendText("Male\t");
checkbox = paragraph.AppendCheckBox();
checkbox.Checked = false;
checkbox.CheckBoxSize = 10;
checkbox.CalculateOnExit = true;
paragraph.AppendText("Female");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();

```

```
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Checkbox.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendText("Gender\t");
//Appends new Checkbox
WCheckBox checkbox = paragraph.AppendCheckBox();
checkbox.Checked = false;
//Sets Checkbox size
checkbox.CheckBoxSize = 10;
checkbox.CalculateOnExit = true;
//Sets help text
checkbox.Help = "Help text";
paragraph.AppendText("Male\t");
checkbox = paragraph.AppendCheckBox();
checkbox.Checked = false;
checkbox.CheckBoxSize = 10;
checkbox.CalculateOnExit = true;
paragraph.AppendText("Female");
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Checkbox.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

You can modify the checkbox properties such as checked state, size, help text in a Word document. The following code illustrates how to modify the checkbox form field properties.

C#

```
//Loads the template document
WordDocument document = new WordDocument("Checkbox.docx");
//Iterates through paragraph items
foreach (ParagraphItem item in document.LastParagraph.ChildEntities)
{
    if (item is WCheckBox)
    {
        WCheckBox checkbox = item as WCheckBox;
```

```

//Modifies check box properties
if (checkbox.Checked)
checkbox.Checked = false;
checkbox.SizeType = CheckBoxSizeType.Exactly;
}
}
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Checkbox.docx")
'Iterates through paragraph items
For Each item As ParagraphItem In document.LastParagraph.ChildEntities
If TypeOf item Is WCheckBox Then
Dim checkbox As WCheckBox = TryCast(item, WCheckBox)
'Modifies check box properties
If checkbox.Checked Then
checkbox.Checked = False
End If
checkbox.SizeType = CheckBoxSizeType.Exactly
End If
Next
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Checkbox.docx"),
FormatType.Docx);
//Iterates through paragraph items
foreach (ParagraphItem item in document.LastParagraph.ChildEntities)
{
if (item is WCheckBox)
{
WCheckBox checkbox = item as WCheckBox;
//Modifies check box properties
if (checkbox.Checked)
checkbox.Checked = false;
checkbox.SizeType = CheckBoxSizeType.Exactly;
}
}
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document

```

```
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Loads the template document
FileStream fileStreamPath = new FileStream("Checkbox.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Iterates through paragraph items
foreach (ParagraphItem item in document.LastParagraph.ChildEntities)
{
    if (item is WCheckBox)
    {
        WCheckBox checkbox = item as WCheckBox;
        //Modifies check box properties
        if (checkbox.Checked)
            checkbox.Checked = false;
        checkbox.SizeType = CheckBoxSizeType.Exactly;
    }
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Checkbo
x.docx"), FormatType.Docx);
//Iterates through paragraph items
foreach (ParagraphItem item in document.LastParagraph.ChildEntities)
{
    if (item is WCheckBox)
    {
        WCheckBox checkbox = item as WCheckBox;
        //Modifies check box properties
        if (checkbox.Checked)
            checkbox.Checked = false;
        checkbox.SizeType = CheckBoxSizeType.Exactly;
    }
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
```



```
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Drop-Down

You can add new Dropdown form field to a Word document by using `AppendDropDownFormField` method of `WParagraph` class.

The following code illustrates how to add a new dropdown field.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendText("Educational Qualification\t");
//Appends Dropdown field
WDropDownFormField dropDownField = paragraph.AppendDropDownFormField();
//Adds items to the Dropdown items collection
dropDownField.DropDownItems.Add("Higher");
dropDownField.DropDownItems.Add("Vocational");
dropDownField.DropDownItems.Add("Universal");
dropDownField.Enabled = true;
//Sets the item index for default value
dropDownField.DropDownSelectedIndex = 1;
dropDownField.CalculateOnExit = true;
//Saves the Word document
document.Save("Dropdown.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As WParagraph = TryCast(section.AddParagraph(), WParagraph)
paragraph.AppendText("Educational Qualification" & vbTab)
'Appends Dropdown field
Dim dropDownField As WDropDownFormField =
paragraph.AppendDropDownFormField()
'Adds items to the Dropdown items collection
dropDownField.DropDownItems.Add("Higher")
dropDownField.DropDownItems.Add("Vocational")
dropDownField.DropDownItems.Add("Universal")
dropDownField.Enabled = True
```

```
'Sets the item index for default value
dropDownField.DropDownSelectedIndex = 1
dropDownField.CalculateOnExit = True
'Saves the Word document
document.Save("Dropdown.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendText("Educational Qualification\t");
//Appends Dropdown field
WDropDownFormField dropDownField = paragraph.AppendDropDownFormField();
//Adds items to the Dropdown items collection
dropDownField.DropDownItems.Add("Higher");
dropDownField.DropDownItems.Add("Vocational");
dropDownField.DropDownItems.Add("Universal");
dropDownField.Enabled = true;
//Sets the item index for default value
dropDownField.DropDownSelectedIndex = 1;
dropDownField.CalculateOnExit = true;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Dropdown.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendText("Educational Qualification\t");
//Appends Dropdown field
WDropDownFormField dropDownField = paragraph.AppendDropDownFormField();
//Adds items to the Dropdown items collection
dropDownField.DropDownItems.Add("Higher");
dropDownField.DropDownItems.Add("Vocational");
dropDownField.DropDownItems.Add("Universal");
dropDownField.Enabled = true;
//Sets the item index for default value
dropDownField.DropDownSelectedIndex = 1;
```

```

dropDownField.CalculateOnExit = true;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Dropdown.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendText("Educational Qualification\t");
//Appends Dropdown field
WDropDownFormField dropDownField = paragraph.AppendDropDownFormField();
//Adds items to the Dropdown items collection
dropDownField.DropDownItems.Add("Higher");
dropDownField.DropDownItems.Add("Vocational");
dropDownField.DropDownItems.Add("Universal");
dropDownField.Enabled = true;
//Sets the item index for default value
dropDownField.DropDownSelectedIndex = 1;
dropDownField.CalculateOnExit = true;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Dropdown.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

You can add or modify list of items of a Dropdown form field in a Word document. The following code illustrates how to modify the dropdown list of a Dropdown form field.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Dropdown.docx");
//Iterates through paragraph items
foreach (ParagraphItem item in document.LastParagraph.ChildEntities)
{
    if (item is WDropDownFormField)
    {
        WDropDownFormField dropdown = item as WDropDownFormField;
        //Modifies the dropdown items
    }
}

```

```

dropdown.DropDownItems.Remove(1);
dropdown.DropDownSelectedIndex = 0;
dropdown.CharacterFormat.FontName = "Arial";
}
}
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Dropdown.docx")
'Iterates through paragraph items
For Each item As ParagraphItem In document.LastParagraph.ChildEntities
If TypeOf item Is WDropDownFormField Then
Dim dropdown As WDropDownFormField = TryCast(item, WDropDownFormField)
'Modifies the dropdown items
dropdown.DropDownItems.Remove(1)
dropdown.DropDownSelectedIndex = 0
dropdown.CharacterFormat.FontName = "Arial"
End If
Next
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Dropdown.docx"),
FormatType.Docx);
//Iterates through paragraph items
foreach (ParagraphItem item in document.LastParagraph.ChildEntities)
{
if (item is WDropDownFormField)
{
WDropDownFormField dropdown = item as WDropDownFormField;
//Modifies the dropdown items
dropdown.DropDownItems.Remove(1);
dropdown.DropDownSelectedIndex = 0;
dropdown.CharacterFormat.FontName = "Arial";
}
}
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform

```

[//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp](https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp)

ASP.NET CORE

```
//Loads the template document
FileStream fileStreamPath = new FileStream("Dropdown.docx", FileMode.Open,
    FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Iterates through paragraph items
foreach (ParagraphItem item in document.LastParagraph.ChildEntities)
{
    if (item is WDropDownFormField)
    {
        WDropDownFormField dropdown = item as WDropDownFormField;
        //Modifies the dropdown items
        dropdown.DropDownItems.Remove(1);
        dropdown.DropDownSelectedIndex = 0;
        dropdown.CharacterFormat.FontName = "Arial";
    }
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
    WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.DropDown
        n.docx"), FormatType.Docx);
//Iterates through paragraph items
foreach (ParagraphItem item in document.LastParagraph.ChildEntities)
{
    if (item is WDropDownFormField)
    {
        WDropDownFormField dropdown = item as WDropDownFormField;
        //Modifies the dropdown items
        dropdown.DropDownItems.Remove(1);
        dropdown.DropDownSelectedIndex = 0;
        dropdown.CharacterFormat.FontName = "Arial";
    }
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
```

```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Text Form field

You can add new text form field to a Word document by using `AppendTextFormField` method of `WParagraph` class.

The following code illustrates how to add new text form field.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendText("General Information");
section.AddParagraph();
paragraph = section.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("Name\t");
text.CharacterFormat.Bold = true;
//Appends Text form field
WTextFormField textField = paragraph.AppendTextFormField(null);
//Sets type of Text form field
textField.Type = TextFormFieldType.RegularText;
textField.CharacterFormat.FontName = "Calibri";
textField.CalculateOnExit = true;
section.AddParagraph();
paragraph = section.AddParagraph() as WParagraph;
text = paragraph.AppendText("Date of Birth\t");
text.CharacterFormat.Bold = true;
//Appends Text form field
textField = paragraph.AppendTextFormField("Date field",
DateTime.Now.ToString("MM/DD/YY"));
textField.StringFormat = "MM/DD/YY";
//Sets Text form field type
textField.Type = TextFormFieldType.DateText;
textField.CalculateOnExit = true;
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
section As IWSection = document.AddSection()
'Adds new paragraph to the section
```

```

Dim paragraph As WParagraph = TryCast(section.AddParagraph(), WParagraph)
paragraph.AppendText("General Information")
section.AddParagraph()
paragraph = TryCast(section.AddParagraph(), WParagraph)
Dim text As ITextRange = paragraph.AppendText("Name" & vbTab)
text.CharacterFormat.Bold = True
'Appends Text form field
Dim textField As WTextFormField = paragraph.AppendTextFormField(Nothing)
'Sets type of Text form field
textField.Type = TextFormFieldType.RegularText
textField.CharacterFormat.FontName = "Calibri"
textField.CalculateOnExit = True
section.AddParagraph()
paragraph = TryCast(section.AddParagraph(), WParagraph)
text = paragraph.AppendText("Date of Birth" & vbTab)
text.CharacterFormat.Bold = True
'Appends Text form field
textField = paragraph.AppendTextFormField("Date field",
DateTime.Now.ToString("MM/DD/YY"))
textField.StringFormat = "MM/DD/YY"
'Sets Text form field type
textField.Type = TextFormFieldType.DateText
textField.CalculateOnExit = True
'Saves the Word document
document.Save("textField.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendText("General Information");
section.AddParagraph();
paragraph = section.AddParagraph() as WParagraph;
ITextRange text = paragraph.AppendText("Name\t");
text.CharacterFormat.Bold = true;
//Appends Text form field
WTextFormField textField = paragraph.AppendTextFormField(null);
//Sets type of Text form field
textField.Type = TextFormFieldType.RegularText;
textField.CharacterFormat.FontName = "Calibri";
textField.CalculateOnExit = true;
section.AddParagraph();
paragraph = section.AddParagraph() as WParagraph;
text = paragraph.AppendText("Date of Birth\t");
text.CharacterFormat.Bold = true;
//Appends Text form field
textField = paragraph.AppendTextFormField("Date field",
DateTime.Now.ToString("MM/DD/YY"));
textField.StringFormat = "MM/DD/YY";
//Sets Text form field type

```

```

textField.Type = TextFormFieldType.DateText;
textField.CalculateOnExit = true;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendText("General Information");
section.AddParagraph();
paragraph = section.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("Name\t");
text.CharacterFormat.Bold = true;
//Appends Text form field
WTextFormField textField = paragraph.AppendTextFormField(null);
//Sets type of Text form field
textField.Type = TextFormFieldType.RegularText;
textField.CharacterFormat.FontName = "Calibri";
textField.CalculateOnExit = true;
section.AddParagraph();
paragraph = section.AddParagraph() as WParagraph;
text = paragraph.AppendText("Date of Birth\t");
text.CharacterFormat.Bold = true;
//Appends Text form field
textField = paragraph.AppendTextFormField("Date field",
DateTime.Now.ToString("MM/DD/YY"));
textField.StringFormat = "MM/DD/YY";
//Sets Text form field type
textField.Type = TextFormFieldType.DateText;
textField.CalculateOnExit = true;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();

```



```

//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
paragraph.AppendText("General Information");
section.AddParagraph();
paragraph = section.AddParagraph() as WParagraph;
IWTextRange text = paragraph.AppendText("Name\t");
text.CharacterFormat.Bold = true;
//Appends Text form field
WTextFormField textField = paragraph.AppendTextFormField(null);
//Sets type of Text form field
textField.Type = TextFormFieldType.RegularText;
textField.CharacterFormat.FontName = "Calibri";
textField.CalculateOnExit = true;
section.AddParagraph();
paragraph = section.AddParagraph() as WParagraph;
text = paragraph.AppendText("Date of Birth\t");
text.CharacterFormat.Bold = true;
//Appends Text form field
textField = paragraph.AppendTextFormField("Date field",
DateTime.Now.ToString("MM/DD/YY"));
textField.StringFormat = "MM/DD/YY";
//Sets Text form field type
textField.Type = TextFormFieldType.DateText;
textField.CalculateOnExit = true;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

You can add or modify text form field properties such as default text, type in a Word document. The following code illustrates how to modify the text form field

C#

```

//Loads the template document
WordDocument document = new WordDocument("Template.docx");
//Iterates through section
foreach (WSection section in document.Sections)
//Iterates through section child elements
foreach (WTextBody textBody in section.ChildEntities)
{
    //Iterates through form fields
    foreach (WFormField formField in textBody.FormFields)
    {
        switch (formField.FormFieldType)
        {

```

```

case FormFieldType.TextInput:
WTextFormField textField = formField as WTextFormField;
if (textField.Type == TextFormFieldType.DateText)
{
    //Modifies the text form field
    textField.Type = TextFormFieldType.RegularText;
    textField.StringFormat = "";
    textField.DefaultText = "Default text";
    textField.Text = "Default text";
    textField.CalculateOnExit = false;
}
break;
}
}
}
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx")
'Iterates through section
For Each section As WSection In document.Sections
'Iterates through section child elements
For Each textBody As WTextBody In section.ChildEntities
'Iterates through form fields
For Each formField As WFormField In textBody.FormFields
Select Case formField.FormFieldType
Case FormFieldType.TextInput
Dim textField As WTextFormField = TryCast(formField, WTextFormField)
If textField.Type == TextFormFieldType.DateText Then
    'Modifies the text form field
    textField.Type = TextFormFieldType.RegularText
    textField.StringFormat = ""
    textField.DefaultText = "Default text"
    textField.Text = "Default text"
    textField.CalculateOnExit = False
End If
Exit Select
End Select
Next
Next
Next
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;

```

```

WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx
"), FormatType.Docx);
//Iterates through section
foreach (WSection section in document.Sections)
//Iterates through section child elements
foreach (WTextBody textBody in section.ChildEntities)
{
//Iterates through form fields
foreach (WFormField formField in textBody.FormFields)
{
switch (formField.FormFieldType)
{
case FormFieldType.TextInput:
WTextFormField textField = formField as WTextFormField;
if (textField.Type == TextFormFieldType.DateText)
{
//Modifies the text form field
textField.Type = TextFormFieldType.RegularText;
textField.StringFormat = "";
textField.DefaultText = "Default text";
textField.Text = "Default text";
textField.CalculateOnExit = false;
}
break;
}
}
}
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads the template document
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Iterates through section
foreach (WSection section in document.Sections)
//Iterates through section child elements
foreach (WTextBody textBody in section.ChildEntities)
{
//Iterates through form fields
foreach (WFormField formField in textBody.FormFields)
{
switch (formField.FormFieldType)
{
case FormFieldType.TextInput:

```

```

WTextFormField textField = formField as WTextFormField;
if (textField.Type == TextFormFieldType.DateText)
{
    //Modifies the text form field
    textField.Type = TextFormFieldType.RegularText;
    textField.StringFormat = "";
    textField.DefaultText = "Default text";
    textField.Text = "Default text";
    textField.CalculateOnExit = false;
}
break;
}
}
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Templat
e.docx"), FormatType.Docx);
//Iterates through section
foreach (WSection section in document.Sections)
//Iterates through section child elements
foreach (WTextBody textBody in section.ChildEntities)
{
    //Iterates through form fields
    foreach (WFormField formField in textBody.FormFields)
    {
        switch (formField.FormFieldType)
        {
            case FormFieldType.TextInput:
                WTextFormField textField = formField as WTextFormField;
                if (textField.Type == TextFormFieldType.DateText)
                {
                    //Modifies the text form field
                    textField.Type = TextFormFieldType.RegularText;
                    textField.StringFormat = "";
                    textField.DefaultText = "Default text";
                    textField.Text = "Default text";
                    textField.CalculateOnExit = false;
                }
                break;
            }
        }
    }
    //Saves the Word document to MemoryStream

```

```

MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Working with Charts

A Chart is a graphical representation of data where the data is represented as symbols such as bars, lines etc. Charts can represent numerical data, functions or some kinds of qualitative structures. DocIO supports the following chart types:

- Bar chart
- Line chart
- Pie chart
- Area chart
- Column chart
- Scatter chart
- Surface chart
- Stock chart
- Radar chart

Note: DocIO provides chart support in following platforms – Windows, ASP.NET, WPF, ASP.NET MVC

DocIO supports chart only in DOCX and WordML format documents.

Creating a simple Chart from scratch

A new chart can be created or an existing chart can be modified by using the WChart instance. The following code example illustrates how to create a new chart.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds section to the document
IWSection sec = document.AddSection();
//Adds paragraph to the section
IWParagraph paragraph = sec.AddParagraph();
//Creates and Appends chart to the paragraph
WChart chart = paragraph.AppendChart(446, 270);
//Sets chart type
chart.ChartType = OfficeChartType.Pie;
//Sets chart title
chart.ChartTitle = "Best Selling Products";
chart.ChartTitleArea.FontName = "Calibri";
chart.ChartTitleArea.Size = 14;
//Sets data for chart
chart.ChartData.SetValue(1, 1, "");

```

```

chart.ChartData.SetValue(1, 2, "Sales");
chart.ChartData.SetValue(2, 1, "Phyllis Lapin");
chart.ChartData.SetValue(2, 2, 141.396);
chart.ChartData.SetValue(3, 1, "Stanley Hudson");
chart.ChartData.SetValue(3, 2, 80.368);
chart.ChartData.SetValue(4, 1, "Bernard Shah");
chart.ChartData.SetValue(4, 2, 71.155);
chart.ChartData.SetValue(5, 1, "Patricia Lincoln");
chart.ChartData.SetValue(5, 2, 47.234);
chart.ChartData.SetValue(6, 1, "Camembert Pierrot");
chart.ChartData.SetValue(6, 2, 46.825);
chart.ChartData.SetValue(7, 1, "Thomas Hardy");
chart.ChartData.SetValue(7, 2, 42.593);
chart.ChartData.SetValue(8, 1, "Hanna Moos");
chart.ChartData.SetValue(8, 2, 41.819);
chart.ChartData.SetValue(9, 1, "Alice Mutton");
chart.ChartData.SetValue(9, 2, 32.698);
chart.ChartData.SetValue(10, 1, "Christina Berglund");
chart.ChartData.SetValue(10, 2, 29.171);
chart.ChartData.SetValue(11, 1, "Elizabeth Lincoln");
chart.ChartData.SetValue(11, 2, 25.696);
//Creates a new chart series with the name "Sales"
IOfficeChartSerie pieSeries = chart.Series.Add("Sales");
pieSeries.Values = chart.ChartData[2, 2, 11, 2];
//Sets data label
pieSeries.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
pieSeries.DataPoints.DefaultDataPoint.DataLabels.Position =
OfficeDataLabelPosition.Outside;
//Sets background color
chart.ChartArea.Fill.ForeColor = Color.FromArgb(242, 242, 242);
chart.PlotArea.Fill.ForeColor = Color.FromArgb(242, 242, 242);
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Sets category labels
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData[2, 1, 11, 1];
//Saves the document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds section to the document
Dim sec As IWSection = document.AddSection()
'Adds paragraph to the section
Dim paragraph As IWParagraph = sec.AddParagraph()
'Creates and Appends chart to the paragraph
Dim chart As WChart = paragraph.AppendChart(446, 270)
'Sets chart type
chart.ChartType = OfficeChartType.Pie
'Sets chart title
chart.ChartTitle = "Best Selling Products"
chart.ChartTitleArea.FontName = "Calibri"
chart.ChartTitleArea.Size = 14
'Sets data for chart

```

```

chart.ChartData.SetValue(1, 1, "")
chart.ChartData.SetValue(1, 2, "Sales")
chart.ChartData.SetValue(2, 1, "Phyllis Lapin")
chart.ChartData.SetValue(2, 2, 141.396)
chart.ChartData.SetValue(3, 1, "Stanley Hudson")
chart.ChartData.SetValue(3, 2, 80.368)
chart.ChartData.SetValue(4, 1, "Bernard Shah")
chart.ChartData.SetValue(4, 2, 71.155)
chart.ChartData.SetValue(5, 1, "Patricia Lincoln")
chart.ChartData.SetValue(5, 2, 47.234)
chart.ChartData.SetValue(6, 1, "Camembert Pierrot")
chart.ChartData.SetValue(6, 2, 46.825)
chart.ChartData.SetValue(7, 1, "Thomas Hardy")
chart.ChartData.SetValue(7, 2, 42.593)
chart.ChartData.SetValue(8, 1, "Hanna Moos")
chart.ChartData.SetValue(8, 2, 41.819)
chart.ChartData.SetValue(9, 1, "Alice Mutton")
chart.ChartData.SetValue(9, 2, 32.698)
chart.ChartData.SetValue(10, 1, "Christina Berglund")
chart.ChartData.SetValue(10, 2, 29.171)
chart.ChartData.SetValue(11, 1, "Elizabeth Lincoln")
chart.ChartData.SetValue(11, 2, 25.696)
'Creates a new chart series with the name "Sales"
Dim pieSeries As IOOfficeChartSerie = chart.Series.Add("Sales")
pieSeries.Values = chart.ChartData(2, 2, 11, 2)
'Sets data label
pieSeries.DataPoints.DefaultDataPoint.DataLabels.IsValue = True
pieSeries.DataPoints.DefaultDataPoint.DataLabels.Position =
OfficeDataLabelPosition.Outside
'Sets background color
chart.ChartArea.Fill.ForeColor = Color.FromArgb(242, 242, 242)
chart.PlotArea.Fill.ForeColor = Color.FromArgb(242, 242, 242)
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None
'Sets category labels
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData(2, 1, 11, 1)
'Saves the document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

using (WordDocument document = new WordDocument())
{
    //Adds section to the document
    IWSection sec = document.AddSection();
    //Adds paragraph to the section
    IWParagraph paragraph = sec.AddParagraph();
    //Creates and Appends chart to the paragraph
    WChart chart = paragraph.AppendChart(446, 270);
    //Sets chart type
    chart.ChartType = OfficeChartType.Pie;
    //Sets chart title
    chart.ChartTitle = "Best Selling Products";
    chart.ChartTitleArea.FontName = "Calibri";
    chart.ChartTitleArea.Size = 14;
}

```

```

//Sets data for chart
chart.ChartData.SetValue(1, 1, "");
chart.ChartData.SetValue(1, 2, "Sales");
chart.ChartData.SetValue(2, 1, "Phyllis Lapin");
chart.ChartData.SetValue(2, 2, 141.396);
chart.ChartData.SetValue(3, 1, "Stanley Hudson");
chart.ChartData.SetValue(3, 2, 80.368);
chart.ChartData.SetValue(4, 1, "Bernard Shah");
chart.ChartData.SetValue(4, 2, 71.155);
chart.ChartData.SetValue(5, 1, "Patricia Lincoln");
chart.ChartData.SetValue(5, 2, 47.234);
chart.ChartData.SetValue(6, 1, "Camembert Pierrot");
chart.ChartData.SetValue(6, 2, 46.825);
chart.ChartData.SetValue(7, 1, "Thomas Hardy");
chart.ChartData.SetValue(7, 2, 42.593);
chart.ChartData.SetValue(8, 1, "Hanna Moos");
chart.ChartData.SetValue(8, 2, 41.819);
chart.ChartData.SetValue(9, 1, "Alice Mutton");
chart.ChartData.SetValue(9, 2, 32.698);
chart.ChartData.SetValue(10, 1, "Christina Berglund");
chart.ChartData.SetValue(10, 2, 29.171);
chart.ChartData.SetValue(11, 1, "Elizabeth Lincoln");
chart.ChartData.SetValue(11, 2, 25.696);
//Creates a new chart series with the name "Sales"
IOfficeChartSerie pieSeries = chart.Series.Add("Sales");
pieSeries.Values = chart.ChartData[2, 2, 11, 2];
//Sets data label
pieSeries.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
pieSeries.DataPoints.DefaultDataPoint.DataLabels.Position =
OfficeDataLabelPosition.Outside;
//Sets background color
chart.ChartArea.Fill.ForeColor = Syncfusion.Drawing.Color.FromArgb(242, 242,
242);
chart.PlotArea.Fill.ForeColor = Syncfusion.Drawing.Color.FromArgb(242, 242,
242);
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Sets category labels
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData[2, 1, 11, 1];
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
}

```

ASP.NET CORE

```

//Creates a new instance of WordDocument (Empty Word Document)
using (WordDocument document = new WordDocument())
{
    //Adds section to the document
    IWSection sec = document.AddSection();
}

```



```

//Adds paragraph to the section
IWParagraph paragraph = sec.AddParagraph();
//Creates and Appends chart to the paragraph
WChart chart = paragraph.AppendChart(446, 270);
//Sets chart type
chart.ChartType = OfficeChartType.Pie;
//Sets chart title
chart.ChartTitle = "Best Selling Products";
chart.ChartTitleArea.FontName = "Calibri";
chart.ChartTitleArea.Size = 14;
//Sets data for chart
chart.ChartData.SetValue(1, 1, "");
chart.ChartData.SetValue(1, 2, "Sales");
chart.ChartData.SetValue(2, 1, "Phyllis Lapin");
chart.ChartData.SetValue(2, 2, 141.396);
chart.ChartData.SetValue(3, 1, "Stanley Hudson");
chart.ChartData.SetValue(3, 2, 80.368);
chart.ChartData.SetValue(4, 1, "Bernard Shah");
chart.ChartData.SetValue(4, 2, 71.155);
chart.ChartData.SetValue(5, 1, "Patricia Lincoln");
chart.ChartData.SetValue(5, 2, 47.234);
chart.ChartData.SetValue(6, 1, "Camembert Pierrot");
chart.ChartData.SetValue(6, 2, 46.825);
chart.ChartData.SetValue(7, 1, "Thomas Hardy");
chart.ChartData.SetValue(7, 2, 42.593);
chart.ChartData.SetValue(8, 1, "Hanna Moos");
chart.ChartData.SetValue(8, 2, 41.819);
chart.ChartData.SetValue(9, 1, "Alice Mutton");
chart.ChartData.SetValue(9, 2, 32.698);
chart.ChartData.SetValue(10, 1, "Christina Berglund");
chart.ChartData.SetValue(10, 2, 29.171);
chart.ChartData.SetValue(11, 1, "Elizabeth Lincoln");
chart.ChartData.SetValue(11, 2, 25.696);
//Creates a new chart series with the name "Sales"
IOfficeChartSerie pieSeries = chart.Series.Add("Sales");
pieSeries.Values = chart.ChartData[2, 2, 11, 2];
//Sets data label
pieSeries.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
pieSeries.DataPoints.DefaultDataPoint.DataLabels.Position =
OfficeDataLabelPosition.Outside;
//Sets background color
chart.ChartArea.Fill.ForeColor = Syncfusion.Drawing.Color.FromArgb(242, 242,
242);
chart.PlotArea.Fill.ForeColor = Syncfusion.Drawing.Color.FromArgb(242, 242,
242);
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Sets category labels
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData[2, 1, 11, 1];
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

using (WordDocument document = new WordDocument())
{
    //Adds section to the document
    IWSection sec = document.AddSection();
    //Adds paragraph to the section
    IWParagraph paragraph = sec.AddParagraph();
    //Creates and Appends chart to the paragraph
    WChart chart = paragraph.AppendChart(446, 270);
    //Sets chart type
    chart.ChartType = OfficeChartType.Pie;
    //Sets chart title
    chart.ChartTitle = "Best Selling Products";
    chart.ChartTitleArea.FontName = "Calibri";
    chart.ChartTitleArea.Size = 14;
    //Sets data for chart
    chart.ChartData.SetValue(1, 1, "");
    chart.ChartData.SetValue(1, 2, "Sales");
    chart.ChartData.SetValue(2, 1, "Phyllis Lapin");
    chart.ChartData.SetValue(2, 2, 141.396);
    chart.ChartData.SetValue(3, 1, "Stanley Hudson");
    chart.ChartData.SetValue(3, 2, 80.368);
    chart.ChartData.SetValue(4, 1, "Bernard Shah");
    chart.ChartData.SetValue(4, 2, 71.155);
    chart.ChartData.SetValue(5, 1, "Patricia Lincoln");
    chart.ChartData.SetValue(5, 2, 47.234);
    chart.ChartData.SetValue(6, 1, "Camembert Pierrot");
    chart.ChartData.SetValue(6, 2, 46.825);
    chart.ChartData.SetValue(7, 1, "Thomas Hardy");
    chart.ChartData.SetValue(7, 2, 42.593);
    chart.ChartData.SetValue(8, 1, "Hanna Moos");
    chart.ChartData.SetValue(8, 2, 41.819);
    chart.ChartData.SetValue(9, 1, "Alice Mutton");
    chart.ChartData.SetValue(9, 2, 32.698);
    chart.ChartData.SetValue(10, 1, "Christina Berglund");
    chart.ChartData.SetValue(10, 2, 29.171);
    chart.ChartData.SetValue(11, 1, "Elizabeth Lincoln");
    chart.ChartData.SetValue(11, 2, 25.696);
    //Creates a new chart series with the name "Sales"
    IOOfficeChartSerie pieSeries = chart.Series.Add("Sales");
    pieSeries.Values = chart.ChartData[2, 2, 11, 2];
    //Sets data label
    pieSeries.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    pieSeries.DataPoints.DefaultDataPoint.DataLabels.Position =
    OfficeDataLabelPosition.Outside;
    //Sets background color
    chart.ChartArea.Fill.ForeColor = Syncfusion.Drawing.Color.FromArgb(242, 242,
    242);
    chart.PlotArea.Fill.ForeColor = Syncfusion.Drawing.Color.FromArgb(242, 242,
    242);
    chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
    //Sets category labels
    chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData[2, 1, 11, 1];
    MemoryStream stream = new MemoryStream();

```

```
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWorddoc.doc
x", "application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
//Closes the documents
document.Close();
}
```

Creating a Chart from Excel file

The chart data can be set through the object array or can be loaded from the excel stream. The specific range of the data from the excel file can be used to set chart data.

The following code example illustrates the chart data loaded from the excel file.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds section to the document
IWSection sec = document.AddSection();
//Adds paragraph to the section
IWParagraph paragraph = sec.AddParagraph();
//Loads the excel file as stream
Stream excelStream = File.OpenRead("Excel_Template.xlsx");
//Creates and Appends chart to the paragraph with excel stream as parameter
WChart chart = paragraph.AppendChart(excelStream, 1, "B2:C6", 470, 300);
//Sets chart type and title
chart.ChartType = OfficeChartType.Column_Clustered;
chart.ChartTitle = "Purchase Details";
chart.ChartTitleArea.FontName = "Calibri";
chart.ChartTitleArea.Size = 14;
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Sets name to chart series
chart.Series[0].Name = "Sum of Purchases";
chart.Series[1].Name = "Sum of Future Expenses";
chart.PrimaryCategoryAxis.Title = "Products";
chart.PrimaryValueAxis.Title = "In Dollars";
//Sets position of legend
chart.Legend.Position = OfficeLegendPosition.Bottom;
//Saves the document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds section to the document
Dim sec As IWSection = document.AddSection()
'Adds paragraph to the section
Dim paragraph As IWParagraph = sec.AddParagraph()
```

```

'Loads the excel file as stream
Dim excelStream As Stream = File.OpenRead("Excel_Template.xlsx")
'Creates and Appends chart to the paragraph with excel stream as parameter
Dim chart As WChart = paragraph.AppendChart(excelStream, 1, "B2:C6", 470,
300)
'Sets chart type and title
chart.ChartType = OfficeChartType.Column_Clustered
chart.ChartTitle = "Purchase Details"
chart.ChartTitleArea.FontName = "Calibri"
chart.ChartTitleArea.Size = 14
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None
'Sets name to chart series
chart.Series(0).Name = "Sum of Purchases"
chart.Series(1).Name = "Sum of Future Expenses"
chart.PrimaryCategoryAxis.Title = "Products"
chart.PrimaryValueAxis.Title = "In Dollars"
'Sets position of legend
chart.Legend.Position = OfficeLegendPosition.Bottom
'Saves the document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

using (WordDocument document = new WordDocument())
{
    //Adds section to the document
    IWSection sec = document.AddSection();
    //Adds paragraph to the section
    IWParagraph paragraph = sec.AddParagraph();
    //Loads the excel file as stream
    Stream excelStream = File.OpenRead("Excel_Template.xlsx");
    //Creates and Appends chart to the paragraph with excel stream as parameter
    WChart chart = paragraph.AppendChart(excelStream, 1, "B2:C6", 470, 300);
    //Sets chart type and title
    chart.ChartType = OfficeChartType.Column_Clustered;
    chart.ChartTitle = "Purchase Details";
    chart.ChartTitleArea.FontName = "Calibri";
    chart.ChartTitleArea.Size = 14;
    chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
    //Sets name to chart series
    chart.Series[0].Name = "Sum of Purchases";
    chart.Series[1].Name = "Sum of Future Expenses";
    chart.PrimaryCategoryAxis.Title = "Products";
    chart.PrimaryValueAxis.Title = "In Dollars";
    //Sets position of legend
    chart.Legend.Position = OfficeLegendPosition.Bottom;
    MemoryStream stream = new MemoryStream();
    //Saves the Word file to MemoryStream
    await document.SaveAsync(stream, FormatType.Docx);
    //Saves the stream as Word file in local machine
    Save(stream, "Result.docx");
    document.Close();
    //Please refer the below link to save Word document in UWP platform

```

```
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
}
```

ASP.NET CORE

```
//Creates a new instance of WordDocument (Empty Word Document)
using (WordDocument document = new WordDocument())
{
    //Adds section to the document
    IWSection sec = document.AddSection();
    //Adds paragraph to the section
    IWParagraph paragraph = sec.AddParagraph();
    //Loads the excel file as stream
    Stream excelStream = File.OpenRead("Excel_Template.xlsx");
    //Creates and Appends chart to the paragraph with excel stream as parameter
    WChart chart = paragraph.AppendChart(excelStream, 1, "B2:C6", 470, 300);
    //Sets chart type and title
    chart.ChartType = OfficeChartType.Column_Clustered;
    chart.ChartTitle = "Purchase Details";
    chart.ChartTitleArea.FontName = "Calibri";
    chart.ChartTitleArea.Size = 14;
    chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
    //Sets name to chart series
    chart.Series[0].Name = "Sum of Purchases";
    chart.Series[1].Name = "Sum of Future Expenses";
    chart.PrimaryCategoryAxis.Title = "Products";
    chart.PrimaryValueAxis.Title = "In Dollars";
    //Sets position of legend
    chart.Legend.Position = OfficeLegendPosition.Bottom;
    MemoryStream stream = new MemoryStream();
    //Saves the Word document to MemoryStream
    document.Save(stream, FormatType.Docx);
    document.Close();
    stream.Position = 0;
    //Download Word document in the browser
    return File(stream, "application/msword", "Result.docx");
}
```

XAMARIN

```
using (WordDocument document = new WordDocument())
{
    //Adds section to the document
    IWSection sec = document.AddSection();
    //Adds paragraph to the section
    IWParagraph paragraph = sec.AddParagraph();
    //Loads the excel file as stream
    Stream excelStream = File.OpenRead("Excel_Template.xlsx");
    //Creates and Appends chart to the paragraph with excel stream as parameter
    WChart chart = paragraph.AppendChart(excelStream, 1, "B2:C6", 470, 300);
    //Sets chart type and title
    chart.ChartType = OfficeChartType.Column_Clustered;
    chart.ChartTitle = "Purchase Details";
    chart.ChartTitleArea.FontName = "Calibri";
    chart.ChartTitleArea.Size = 14;
}
```

```

chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Sets name to chart series
chart.Series[0].Name = "Sum of Purchases";
chart.Series[1].Name = "Sum of Future Expenses";
chart.PrimaryCategoryAxis.Title = "Products";
chart.PrimaryValueAxis.Title = "In Dollars";
//Sets position of legend
chart.Legend.Position = OfficeLegendPosition.Bottom;
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWorddoc.doc
x", "application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
//Closes the documents
document.Close();
}

```

Creating Custom Charts

A chart is composed of data series. Each data series is represented by a series object. When creating a custom chart, you can specify different series types for each data series.

The following code example illustrates how to create custom charts.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds section to the document
IWSection sec = document.AddSection();
//Adds paragraph to the section
IWParagraph paragraph = sec.AddParagraph();
//Inputs data for chart
object[][] data = new object[6][];
for (int i = 0; i < 6; i++)
data[i] = new object[3];
data[0][0] = "";
data[1][0] = "Camembert Pierrot";
data[2][0] = "Alice Mutton";
data[3][0] = "Roasted Tigers";
data[4][0] = "Orange Shake";
data[5][0] = "Dried Apples";
data[0][1] = "Sum of Purchases";
data[1][1] = 286;
data[2][1] = 680;
data[3][1] = 288;
data[4][1] = 200;
data[5][1] = 731;
data[0][2] = "Sum of Future Expenses";
data[1][2] = 1300;
data[2][2] = 700;
data[3][2] = 1280;
data[4][2] = 1200;

```

```

data[5][2] = 2660;
//Creates and Appends chart to the paragraph
WChart chart = paragraph.AppendChart(data, 470, 300);
//Sets chart type and title
chart.ChartTitle = "Purchase Details";
chart.ChartTitleArea.FontName = "Calibri";
chart.ChartTitleArea.Size = 14;
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Sets series type
chart.Series[0].SeriesType = OfficeChartType.Line_Markers;
chart.Series[1].SeriesType = OfficeChartType.Bar_Clustered;
chart.PrimaryCategoryAxis.Title = "Products";
chart.PrimaryValueAxis.Title = "In Dollars";
//Sets position of legend
chart.Legend.Position = OfficeLegendPosition.Bottom;
//Saves the document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds section to the document
Dim sec As IWSection = document.AddSection()
'Adds paragraph to the section
Dim paragraph As IWParagraph = sec.AddParagraph()
'Inputs data for chart
Dim data As Object()() = New Object(5)() {}
For i As Integer = 0 To 5
    data(i) = New Object(2) {}
Next
data(0)(0) = ""
data(1)(0) = "Camembert Pierrot"
data(2)(0) = "Alice Mutton"
data(3)(0) = "Roasted Tigers"
data(4)(0) = "Orange Shake"
data(5)(0) = "Dried Apples"
data(0)(1) = "Sum of Purchases"
data(1)(1) = 286
data(2)(1) = 680
data(3)(1) = 288
data(4)(1) = 200
data(5)(1) = 731
data(0)(2) = "Sum of Future Expenses"
data(1)(2) = 1300
data(2)(2) = 700
data(3)(2) = 1280
data(4)(2) = 1200
data(5)(2) = 2660
'Creates and Appends chart to the paragraph
Dim chart As WChart = paragraph.AppendChart(data, 470, 300)
'Sets chart type and title
chart.ChartTitle = "Purchase Details"
chart.ChartTitleArea.FontName = "Calibri"

```

```

chart.ChartTitleArea.Size = 14
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None
'Sets series type
chart.Series(0).SerieType = OfficeChartType.Line_Markers
chart.Series(1).SerieType = OfficeChartType.Bar_Clustered
chart.PrimaryCategoryAxis.Title = "Products"
chart.PrimaryValueAxis.Title = "In Dollars"
'Sets position of legend
chart.Legend.Position = OfficeLegendPosition.Bottom
'Saves the document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

using (WordDocument document = new WordDocument())
{
    //Adds section to the document
    IWSection sec = document.AddSection();
    //Adds paragraph to the section
    IWParagraph paragraph = sec.AddParagraph();
    //Inputs data for chart
    object[][] data = new object[6][];
    for (int i = 0; i < 6; i++)
        data[i] = new object[3];
    data[0][0] = "";
    data[1][0] = "Camembert Pierrot";
    data[2][0] = "Alice Mutton";
    data[3][0] = "Roasted Tigers";
    data[4][0] = "Orange Shake";
    data[5][0] = "Dried Apples";
    data[0][1] = "Sum of Purchases";
    data[1][1] = 286;
    data[2][1] = 680;
    data[3][1] = 288;
    data[4][1] = 200;
    data[5][1] = 731;
    data[0][2] = "Sum of Future Expenses";
    data[1][2] = 1300;
    data[2][2] = 700;
    data[3][2] = 1280;
    data[4][2] = 1200;
    data[5][2] = 2660;
    //Creates and Appends chart to the paragraph
    WChart chart = paragraph.AppendChart(data, 470, 300);
    //Sets chart type and title
    chart.ChartTitle = "Purchase Details";
    chart.ChartTitleArea.FontName = "Calibri";
    chart.ChartTitleArea.Size = 14;
    chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
    //Sets series type
    chart.Series[0].SerieType = OfficeChartType.Line_Markers;
    chart.Series[1].SerieType = OfficeChartType.Bar_Clustered;
    chart.PrimaryCategoryAxis.Title = "Products";
    chart.PrimaryValueAxis.Title = "In Dollars";
}

```



```
//Sets position of legend
chart.Legend.Position = OfficeLegendPosition.Bottom;
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
}
```

ASP.NET CORE

```
//Creates a new instance of WordDocument (Empty Word Document)
using (WordDocument document = new WordDocument())
{
    //Adds section to the document
    IWSection sec = document.AddSection();
    //Adds paragraph to the section
    IWParagraph paragraph = sec.AddParagraph();
    //Inputs data for chart
    object[][] data = new object[6][];
    for (int i = 0; i < 6; i++)
    {
        data[i] = new object[3];
        data[0][0] = "";
        data[1][0] = "Camembert Pierrot";
        data[2][0] = "Alice Mutton";
        data[3][0] = "Roasted Tigers";
        data[4][0] = "Orange Shake";
        data[5][0] = "Dried Apples";
        data[0][1] = "Sum of Purchases";
        data[1][1] = 286;
        data[2][1] = 680;
        data[3][1] = 288;
        data[4][1] = 200;
        data[5][1] = 731;
        data[0][2] = "Sum of Future Expenses";
        data[1][2] = 1300;
        data[2][2] = 700;
        data[3][2] = 1280;
        data[4][2] = 1200;
        data[5][2] = 2660;
    }
    //Creates and Appends chart to the paragraph
    WChart chart = paragraph.AppendChart(data, 470, 300);
    //Sets chart type and title
    chart.ChartTitle = "Purchase Details";
    chart.ChartTitleArea.FontName = "Calibri";
    chart.ChartTitleArea.Size = 14;
    chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
    //Sets series type
    chart.Series[0].SeriesType = OfficeChartType.Line_Markers;
    chart.Series[1].SeriesType = OfficeChartType.Bar_Clustered;
    chart.PrimaryCategoryAxis.Title = "Products";
    chart.PrimaryValueAxis.Title = "In Dollars";
}
```

```
//Sets position of legend
chart.Legend.Position = OfficeLegendPosition.Bottom;
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}
```

XAMARIN

```
using (WordDocument document = new WordDocument())
{
    //Adds section to the document
    IWSection sec = document.AddSection();
    //Adds paragraph to the section
    IWParagraph paragraph = sec.AddParagraph();
    //Inputs data for chart
    object[][] data = new object[6][];
    for (int i = 0; i < 6; i++)
        data[i] = new object[3];
    data[0][0] = "";
    data[1][0] = "Camembert Pierrot";
    data[2][0] = "Alice Mutton";
    data[3][0] = "Roasted Tigers";
    data[4][0] = "Orange Shake";
    data[5][0] = "Dried Apples";
    data[0][1] = "Sum of Purchases";
    data[1][1] = 286;
    data[2][1] = 680;
    data[3][1] = 288;
    data[4][1] = 200;
    data[5][1] = 731;
    data[0][2] = "Sum of Future Expenses";
    data[1][2] = 1300;
    data[2][2] = 700;
    data[3][2] = 1280;
    data[4][2] = 1200;
    data[5][2] = 2660;
    //Creates and Appends chart to the paragraph
    WChart chart = paragraph.AppendChart(data, 470, 300);
    //Sets chart type and title
    chart.ChartTitle = "Purchase Details";
    chart.ChartTitleArea.FontName = "Calibri";
    chart.ChartTitleArea.Size = 14;
    chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
    //Sets series type
    chart.Series[0].SerieType = OfficeChartType.Line_Markers;
    chart.Series[1].SerieType = OfficeChartType.Bar_Clustered;
    chart.PrimaryCategoryAxis.Title = "Products";
    chart.PrimaryValueAxis.Title = "In Dollars";
    //Sets position of legend
    chart.Legend.Position = OfficeLegendPosition.Bottom;
    MemoryStream stream = new MemoryStream();
}
```

```
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWorddoc.doc
x", "application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
//Closes the documents
document.Close();
}
```

Refreshing the Chart

The chart may not have the data in the referred excel file instead it may represent some other data. For those charts to have the excel data, it should be refreshed.

The following code example illustrates how to refresh the chart.

C#

```
//Loads the template document
WordDocument document = new WordDocument("Template.docx");
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity from the paragraph items
WChart chart = paragraph.ChildEntities[1] as WChart;
//Refreshes chart data
chart.Refresh();
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Loads the template document
Dim document As New WordDocument("Template.docx")
'Gets the last paragraph
Dim paragraph As WParagraph = document.LastParagraph
'Gets the chart entity from the paragraph items
Dim chart As WChart = TryCast(paragraph.ChildEntities(1), WChart)
'Refreshes chart data
chart.Refresh()
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new
WordDocument((assembly.GetManifestResourceStream("CreateWordSample.Assets.Te
st.docx"))),
```

```

FormatType.Docx))
{
    //Gets the last paragraph
    WParagraph paragraph = document.LastParagraph;
    //Gets the chart entity from the paragraph items
    WChart chart = paragraph.ChildEntities[1] as WChart;
    //Refreshes chart data
    chart.Refresh();
    MemoryStream stream = new MemoryStream();
    await document.SaveAsync(stream, FormatType.Docx);
    //Saves the stream as Word file in local machine
    Save(stream, "Protection.docx");
    //Closes the Word document
    document.Close();
    //Please refer the below link to save Word document in UWP platform
    //https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
}

```

ASP.NET CORE

```

FileStream fileStreamPath = new FileStream(@"Data/Template.docx",
    FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(fileStreamPath,
    FormatType.Docx))
{
    //Gets the last paragraph
    WParagraph paragraph = document.LastParagraph;
    //Gets the chart entity from the paragraph items
    WChart chart = paragraph.ChildEntities[1] as WChart;
    //Refreshes chart data
    chart.Refresh();
    MemoryStream stream = new MemoryStream();
    document.Save(stream, FormatType.docx);
    //Closes the Word document
    document.Close();
    stream.Position = 0;
    //Download Word document in the browser
    return File(stream, "application/msword", "Protection.docx");
}

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new
    WordDocument((assembly.GetManifestResourceStream("XamarinFormsAppl.Assets.He
llo World.docx")),
    FormatType.Docx))
{
    //Gets the last paragraph
    WParagraph paragraph = document.LastParagraph;
    //Gets the chart entity from the paragraph items

```

```

WChart chart = paragraph.ChildEntities[1] as WChart;
//Refreshes chart data
chart.Refresh();
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Protection.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
//Closes the Word document
document.Close();
}

```

Modifying the Chart data

The following code example illustrates how to modify an existing chart data.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Template.docx");
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity from the paragraph items
WChart chart = paragraph.ChildEntities[0] as WChart;
//Modifies the data values of chart
chart.ChartData.SetValue(2, 2, 120);
chart.ChartData.SetValue(3, 2, 60);
chart.ChartData.SetValue(4, 2, 70);
//Refreshes chart data to set the modified values
chart.Refresh();
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx")
'Gets the last paragraph
Dim paragraph As WParagraph = document.LastParagraph
'Gets the chart entity from the paragraph items
Dim chart As WChart = TryCast(paragraph.ChildEntities(0), WChart)
'Modifies the data values of chart
chart.ChartData.SetValue(2, 2, 120)
chart.ChartData.SetValue(3, 2, 60)
chart.ChartData.SetValue(4, 2, 70)
'Refreshes chart data to set the modified values
chart.Refresh()
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new
WordDocument((assembly.GetManifestResourceStream("CreateWordSample.Assets.Te
st.docx")),
FormatType.Docx))
{
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity from the paragraph items
WChart chart = paragraph.ChildEntities[0] as WChart;
//Modifies the data values of chart
chart.ChartData.SetValue(2, 2, 120);
chart.ChartData.SetValue(3, 2, 60);
chart.ChartData.SetValue(4, 2, 70);
//Refreshes chart data to set the modified values
chart.Refresh();
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Closes the Word document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
}
```

ASP.NET CORE

```
FileStream fileStreamPath = new FileStream(@"Data/Template.docx",
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(fileStreamPath,
FormatType.Docx))
{
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity from the paragraph items
WChart chart = paragraph.ChildEntities[0] as WChart;
//Modifies the data values of chart
chart.ChartData.SetValue(2, 2, 120);
chart.ChartData.SetValue(3, 2, 60);
chart.ChartData.SetValue(4, 2, 70);
//Refreshes chart data to set the modified values
chart.Refresh();
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.docx);
//Closes the Word document
document.Close();
stream.Position = 0;
```

```
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}
```

XAMARIN

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new
WordDocument((assembly.GetManifestResourceStream("XamarinFormsAppl.Assets.He
llo World.docx")),
FormatType.Docx))
{
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity from the paragraph items
WChart chart = paragraph.ChildEntities[0] as WChart;
//Modifies the data values of chart
chart.ChartData.SetValue(2, 2, 120);
chart.ChartData.SetValue(3, 2, 60);
chart.ChartData.SetValue(4, 2, 70);
//Refreshes chart data to set the modified values
chart.Refresh();
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
//Closes the Word document
document.Close();
}
```

Customizing the Chart & its elements

A Chart is composed of various elements such as plot area, chart area, title area, legend, data labels, axis etc. The following screenshot illustrates the various elements of chart.



1. The chart area of the chart.
2. The plot area of the chart.
3. The data points of the data series that are plotted in the chart.
4. The horizontal (category) and vertical (value) axis along where the data is plotted in the chart.
5. The legend of the chart.
6. The chart title and axis.
7. A data label that you can use to identify the details of a data point in a data series.

Modifying Chart Appearance

The following code example illustrates how to modify the appearance of an existing chart in the document.

C#

```
//Loads the template document
WordDocument document = new WordDocument("Template.docx", FormatType.Docx);
//Gets the paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity
WChart chart = paragraph.ChildEntities[0] as WChart;
//Modifies the chart height and width
chart.Height = 300;
chart.Width = 500;
//Changes the title
chart.ChartTitle = "New title";
//Changes the series name of first chart series
chart.Series[0].Name = "Modified series name";
//Hides the category labels
chart.CategoryLabelLevel =
OfficeCategoriesLabelLevel.CategoriesLabelLevelNone;
//Shows data Table.
chart.HasDataTable = true;
//Formats the chart area.
IOfficeChartFrameFormat chartArea = chart.ChartArea;
```



```

//Sets border line pattern, color, line weight
chartArea.Border.LinePattern = OfficeChartLinePattern.Solid;
chartArea.Border.LineColor = Color.Blue;
chartArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Sets fill type and fill colors
chartArea.Fill.FillType = OfficeFillType.Gradient;
chartArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
chartArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartArea.Fill.ForeColor = Color.White;
//Plots Area
IOfficeChartFrameFormat chartPlotArea = chart.PlotArea;
//Plots area border settings - line pattern, color, weight
chartPlotArea.Border.LinePattern = OfficeChartLinePattern.Solid;
chartPlotArea.Border.LineColor = Color.Blue;
chartPlotArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Sets fill type and color
chartPlotArea.Fill.FillType = OfficeFillType.Gradient;
chartPlotArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
chartPlotArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartPlotArea.Fill.ForeColor = Color.White;
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx", FormatType.Docx)
'Gets the paragraph
Dim paragraph As WParagraph = document.LastParagraph
'Gets the chart entity
Dim chart As WChart = TryCast(paragraph.ChildEntities(0), WChart)
'Modifies the chart height and width
chart.Height = 300
chart.Width = 500
'Changes the title
chart.ChartTitle = "New title"
'Changes the series name of first chart series
chart.Series(0).Name = "Modified series name"
'Hides the category labels
chart.CategoryLabelLevel =
OfficeCategoriesLabelLevel.CategoriesLabelLevelNone
'Shows data Table.
chart.HasDataTable = True
'Formats chart area.
Dim chartArea As IOfficeChartFrameFormat = chart.ChartArea
'Sets border line pattern, color, line weight
chartArea.Border.LinePattern = OfficeChartLinePattern.Solid
chartArea.Border.LineColor = Color.Blue
chartArea.Border.LineWeight = OfficeChartLineWeight.Hairline
'Sets fill type and fill colors
chartArea.Fill.FillType = OfficeFillType.Gradient
chartArea.Fill.GradientColorType = OfficeGradientColor.TwoColor
chartArea.Fill.BackColor = Color.FromArgb(205, 217, 234)
chartArea.Fill.ForeColor = Color.White
'Plots the Area

```

```

Dim chartPlotArea As IOfficeChartFrameFormat = chart.PlotArea
'Plots area border settings - line pattern, color, weight
chartPlotArea.Border.LinePattern = OfficeChartLinePattern.Solid
chartPlotArea.Border.LineColor = Color.Blue
chartPlotArea.Border.LineWeight = OfficeChartLineWeight.Hairline
'Sets fill type and color
chartPlotArea.Fill.FillType = OfficeFillType.Gradient
chartPlotArea.Fill.GradientColorType = OfficeGradientColor.TwoColor
chartPlotArea.Fill.BackColor = Color.FromArgb(205, 217, 234)
chartPlotArea.Fill.ForeColor = Color.White
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
// Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new
WordDocument((assembly.GetManifestResourceStream("CreateWordSample.Assets.Te
st.docx")),
FormatType.Docx))
{
//Gets the paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity
WChart chart = paragraph.ChildEntities[0] as WChart;
//Modifies the chart height and width
chart.Height = 300;
chart.Width = 500;
//Changes the title
chart.ChartTitle = "New title";
//Changes the series name of first chart series
chart.Series[0].Name = "Modified series name";
//Hides the category labels
chart.CategoryLabelLevel =
OfficeCategoriesLabelLevel.CategoriesLabelLevelNone;
//Shows data Table.
chart.HasDataTable = true;
//Formats the chart area.
IOfficeChartFrameFormat chartArea = chart.ChartArea;
//Sets border line pattern, color, line weight
chartArea.Border.LinePattern = OfficeChartLinePattern.Solid;
chartArea.Border.LineColor = Syncfusion.Drawing.Color.Blue;
chartArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Sets fill type and fill colors
chartArea.Fill.FillType = OfficeFillType.Gradient;
chartArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
chartArea.Fill.BackColor = Syncfusion.Drawing.Color.FromArgb(205, 217, 234);
chartArea.Fill.ForeColor = Syncfusion.Drawing.Color.White;
//Plots Area
IOfficeChartFrameFormat chartPlotArea = chart.PlotArea;
//Plots area border settings - line pattern, color, weight
chartPlotArea.Border.LinePattern = OfficeChartLinePattern.Solid;

```

```

chartPlotArea.Border.LineColor = Syncfusion.Drawing.Color.Blue;
chartPlotArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Sets fill type and color
chartPlotArea.Fill.FillType = OfficeFillType.Gradient;
chartPlotArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
chartPlotArea.Fill.BackColor = Syncfusion.Drawing.Color.FromArgb(205, 217,
234);
chartPlotArea.Fill.ForeColor = Syncfusion.Drawing.Color.White;
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Closes the Word document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
}

```

ASP.NET CORE

```

FileStream fileStreamPath = new FileStream(@"Data/Template.docx",
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(fileStreamPath,
FormatType.Docx))
{
//Gets the paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity
WChart chart = paragraph.ChildEntities[0] as WChart;
//Modifies the chart height and width
chart.Height = 300;
chart.Width = 500;
//Changes the title
chart.ChartTitle = "New title";
//Changes the series name of first chart series
chart.Series[0].Name = "Modified series name";
//Hides the category labels
chart.CategoryLabelLevel =
OfficeCategoriesLabelLevel.CategoriesLabelLevelNone;
//Shows data Table.
chart.HasDataTable = true;
//Formats the chart area.
IOfficeChartFrameFormat chartArea = chart.ChartArea;
//Sets border line pattern, color, line weight
chartArea.Border.LinePattern = OfficeChartLinePattern.Solid;
chartArea.Border.LineColor = Syncfusion.Drawing.Color.Blue;
chartArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Sets fill type and fill colors
chartArea.Fill.FillType = OfficeFillType.Gradient;
chartArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
chartArea.Fill.BackColor = Syncfusion.Drawing.Color.FromArgb(205, 217, 234);
chartArea.Fill.ForeColor = Syncfusion.Drawing.Color.White;
//Plots Area

```

```

IOfficeChartFrameFormat chartPlotArea = chart.PlotArea;
//Plots area border settings - line pattern, color, weight
chartPlotArea.Border.LinePattern = OfficeChartLinePattern.Solid;
chartPlotArea.Border.LineColor = Syncfusion.Drawing.Color.Blue;
chartPlotArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Sets fill type and color
chartPlotArea.Fill.FillType = OfficeFillType.Gradient;
chartPlotArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
chartPlotArea.Fill.BackColor = Syncfusion.Drawing.Color.FromArgb(205, 217,
234);
chartPlotArea.Fill.ForeColor = Syncfusion.Drawing.Color.White;
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.docx);
//Closes the Word document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new
WordDocument((assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.He
llo World.docx")),
FormatType.Docx))
{
//Gets the paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity
WChart chart = paragraph.ChildEntities[0] as WChart;
//Modifies the chart height and width
chart.Height = 300;
chart.Width = 500;
//Changes the title
chart.ChartTitle = "New title";
//Changes the series name of first chart series
chart.Series[0].Name = "Modified series name";
//Hides the category labels
chart.CategoryLabelLevel =
OfficeCategoriesLabelLevel.CategoriesLabelLevelNone;
//Shows data Table.
chart.HasDataTable = true;
//Formats the chart area.
IOfficeChartFrameFormat chartArea = chart.ChartArea;
//Sets border line pattern, color, line weight
chartArea.Border.LinePattern = OfficeChartLinePattern.Solid;
chartArea.Border.LineColor = Syncfusion.Drawing.Color.Blue;
chartArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Sets fill type and fill colors
chartArea.Fill.FillType = OfficeFillType.Gradient;
chartArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
chartArea.Fill.BackColor = Syncfusion.Drawing.Color.FromArgb(205, 217, 234);

```

```

chartArea.Fill.ForeColor = Syncfusion.Drawing.Color.White;
//Plots Area
IOfficeChartFrameFormat chartPlotArea = chart.PlotArea;
//Plots area border settings - line pattern, color, weight
chartPlotArea.Border.LinePattern = OfficeChartLinePattern.Solid;
chartPlotArea.Border.LineColor = Syncfusion.Drawing.Color.Blue;
chartPlotArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Sets fill type and color
chartPlotArea.Fill.FillType = OfficeFillType.Gradient;
chartPlotArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
chartPlotArea.Fill.BackColor = Syncfusion.Drawing.Color.FromArgb(205, 217,
234);
chartPlotArea.Fill.ForeColor = Syncfusion.Drawing.Color.White;
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
//Closes the Word document
document.Close();
}

```

Modifying Plot Area and Legend

The following code example illustrates how to modify the plot area and legend of the chart.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Template.docx", FormatType.Docx);
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity
WChart chart = paragraph.ChildEntities[0] as WChart;
//Sets border settings - line color, pattern, weight, transparency
chart.PlotArea.Border.AutoFormat = false;
chart.PlotArea.Border.IsAutoLineColor = false;
chart.PlotArea.Border.LineColor = Color.Blue;
chart.PlotArea.Border.LinePattern = OfficeChartLinePattern.DashDot;
chart.PlotArea.Border.LineWeight = OfficeChartLineWeight.Wide;
chart.PlotArea.Border.Transparency = 0.6;
//Sets the plot area's fill type, color
chart.PlotArea.Fill.FillType = OfficeFillType.SolidColor;
chart.PlotArea.Fill.ForeColor = Color.LightPink;
//Sets the plot area shadow presence
chart.PlotArea.Shadow.ShadowInnerPresets =
Office2007ChartPresetsInner.InsideDiagonalTopLeft;
//Sets the legend position
chart.Legend.Position = OfficeLegendPosition.Left;
//Sets the layout inclusion
chart.Legend.IncludeInLayout = true;
//Sets the legend border format - color, pattern, weight
chart.Legend.FrameFormat.Border.AutoFormat = false;
chart.Legend.FrameFormat.Border.IsAutoLineColor = false;

```

```

chart.Legend.FrameFormat.Border.LineColor = Color.Blue;
chart.Legend.FrameFormat.Border.LinePattern =
OfficeChartLinePattern.DashDot;
chart.Legend.FrameFormat.Border.LineWeight = OfficeChartLineWeight.Wide;
//Sets the legend's text area formatting - font name, weight, color, size
chart.Legend.TextArea.Bold = true;
chart.Legend.TextArea.Color = OfficeKnownColors.Bright_green;
chart.Legend.TextArea.FontName = "Times New Roman";
chart.Legend.TextArea.Size = 20;
chart.Legend.TextArea.Strikethrough = true;
//Modifies the legend entry
chart.Legend.LegendEntries[0].IsDeleted = true;
//Modifies the legend layout - height, left, top, width
chart.Legend.Layout.Height = 50;
chart.Legend.Layout.HeightMode = LayoutModes.factor;
chart.Legend.Layout.Left = 10;
chart.Legend.Layout.LeftMode = LayoutModes.factor;
chart.Legend.Layout.Top = 30;
chart.Legend.Layout.TopMode = LayoutModes.factor;
chart.Legend.Layout.Width = 100;
chart.Legend.Layout.WidthMode = LayoutModes.factor;
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx", FormatType.Docx)
Dim paragraph As WParagraph = document.LastParagraph
'Gets the chart entity
Dim chart As WChart = TryCast(paragraph.ChildEntities(0), WChart)
'Set border settings - line color, pattern, weight, transparency
chart.PlotArea.Border.AutoFormat = False
chart.PlotArea.Border.IsAutoLineColor = False
chart.PlotArea.Border.LineColor = Color.Blue
chart.PlotArea.Border.LinePattern = OfficeChartLinePattern.DashDot
chart.PlotArea.Border.LineWeight = OfficeChartLineWeight.Wide
chart.PlotArea.Border.Transparency = 0.6
'Sets the plot area's fill type, color
chart.PlotArea.Fill.FillType = OfficeFillType.SolidColor
chart.PlotArea.Fill.ForeColor = Color.LightPink
'Sets the plot area shadow presence
chart.PlotArea.Shadow.ShadowInnerPresets =
Office2007ChartPresetsInner.InsideDiagonalTopLeft
'Sets the legend position
chart.Legend.Position = OfficeLegendPosition.Left
'Sets the layout inclusion
chart.Legend.IncludeInLayout = True
'Sets the legend border format - color, pattern, weight
chart.Legend.FrameFormat.Border.AutoFormat = False
chart.Legend.FrameFormat.Border.IsAutoLineColor = False
chart.Legend.FrameFormat.Border.LineColor = Color.Blue
chart.Legend.FrameFormat.Border.LinePattern = OfficeChartLinePattern.DashDot
chart.Legend.FrameFormat.Border.LineWeight = OfficeChartLineWeight.Wide
'Sets the legend's text area formatting - font name, weight, color, size

```

```

chart.Legend.TextArea.Bold = True
chart.Legend.TextArea.Color = OfficeKnownColors.Bright_green
chart.Legend.TextArea.FontName = "Times New Roman"
chart.Legend.TextArea.Size = 20
chart.Legend.TextArea.Strikethrough = True
'Modifies the legend entry
chart.Legend.LegendEntries(0).IsDeleted = True
'Modifies the legend layout - height, left, top, width
chart.Legend.Layout.Height = 50
chart.Legend.Layout.HeightMode = LayoutModes.factor
chart.Legend.Layout.Left = 10
chart.Legend.Layout.LeftMode = LayoutModes.factor
chart.Legend.Layout.Top = 30
chart.Legend.Layout.TopMode = LayoutModes.factor
chart.Legend.Layout.Width = 100
chart.Legend.Layout.WidthMode = LayoutModes.factor
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new
WordDocument((assembly.GetManifestResourceStream("CreateWordSample.Assets.Te
st.docx")),
FormatType.Docx)
{
//Gets the paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity
WChart chart = paragraph.ChildEntities[0] as WChart;
//Sets border settings - line color, pattern, weight, transparency
chart.PlotArea.Border.AutoFormat = false;
chart.PlotArea.Border.IsAutoLineColor = false;
chart.PlotArea.Border.LineColor = Syncfusion.Drawing.Color.Blue;
chart.PlotArea.Border.LinePattern = OfficeChartLinePattern.DashDot;
chart.PlotArea.Border.LineWeight = OfficeChartLineWeight.Wide;
chart.PlotArea.Border.Transparency = 0.6;
//Sets the plot area's fill type, color
chart.PlotArea.Fill.FillType = OfficeFillType.SolidColor;
chart.PlotArea.Fill.ForeColor = Syncfusion.Drawing.Color.LightPink;
//Sets the plot area shadow presence
chart.PlotArea.Shadow.ShadowInnerPresets =
Office2007ChartPresetsInner.InsideDiagonalTopLeft;
//Sets the legend position
chart.Legend.Position = OfficeLegendPosition.Left;
//Sets the layout inclusion
chart.Legend.IncludeInLayout = true;
//Sets the legend border format - color, pattern, weight
chart.Legend.FrameFormat.Border.AutoFormat = false;
chart.Legend.FrameFormat.Border.IsAutoLineColor = false;
chart.Legend.FrameFormat.Border.LineColor = Syncfusion.Drawing.Color.Blue;

```

```

chart.Legend.FrameFormat.Border.LinePattern =
OfficeChartLinePattern.DashDot;
chart.Legend.FrameFormat.Border.LineWeight = OfficeChartLineWeight.Wide;
//Sets the legend's text area formatting - font name, weight, color, size
chart.Legend.TextArea.Bold = true;
chart.Legend.TextArea.Color = OfficeKnownColors.Bright_green;
chart.Legend.TextArea.FontName = "Times New Roman";
chart.Legend.TextArea.Size = 20;
chart.Legend.TextArea.Strikethrough = true;
//Modifies the legend entry
chart.Legend.LegendEntries[0].IsDeleted = true;
//Modifies the legend layout - height, left, top, width
chart.Legend.Layout.Height = 50;
chart.Legend.Layout.HeightMode = LayoutModes.factor;
chart.Legend.Layout.Left = 10;
chart.Legend.Layout.LeftMode = LayoutModes.factor;
chart.Legend.Layout.Top = 30;
chart.Legend.Layout.TopMode = LayoutModes.factor;
chart.Legend.Layout.Width = 100;
chart.Legend.Layout.WidthMode = LayoutModes.factor;
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
//Closes the Word document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
}

```

ASP.NET CORE

```

FileStream fileStreamPath = new FileStream(@"Data/Template.docx",
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(fileStreamPath,
FormatType.Docx))
{
//Gets the paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity
WChart chart = paragraph.ChildEntities[0] as WChart;
//Sets border settings - line color, pattern, weight, transparency
chart.PlotArea.Border.AutoFormat = false;
chart.PlotArea.Border.IsAutoLineColor = false;
chart.PlotArea.Border.LineColor = Syncfusion.Drawing.Color.Blue;
chart.PlotArea.Border.LinePattern = OfficeChartLinePattern.DashDot;
chart.PlotArea.Border.LineWeight = OfficeChartLineWeight.Wide;
chart.PlotArea.Border.Transparency = 0.6;
//Sets the plot area's fill type, color
chart.PlotArea.Fill.FillType = OfficeFillType.SolidColor;
chart.PlotArea.Fill.ForeColor = Syncfusion.Drawing.Color.LightPink;
//Sets the plot area shadow presence

```



```

chart.PlotArea.Shadow.ShadowInnerPresets =
Office2007ChartPresetsInner.InsideDiagonalTopLeft;
//Sets the legend position
chart.Legend.Position = OfficeLegendPosition.Left;
//Sets the layout inclusion
chart.Legend.IncludeInLayout = true;
//Sets the legend border format - color, pattern, weight
chart.Legend.FrameFormat.Border.AutoFormat = false;
chart.Legend.FrameFormat.Border.IsAutoLineColor = false;
chart.Legend.FrameFormat.Border.LineColor = Syncfusion.Drawing.Color.Blue;
chart.Legend.FrameFormat.Border.LinePattern =
OfficeChartLinePattern.DashDot;
chart.Legend.FrameFormat.Border.LineWeight = OfficeChartLineWeight.Wide;
//Sets the legend's text area formatting - font name, weight, color, size
chart.Legend.TextArea.Bold = true;
chart.Legend.TextArea.Color = OfficeKnownColors.Bright_green;
chart.Legend.TextArea.FontName = "Times New Roman";
chart.Legend.TextArea.Size = 20;
chart.Legend.TextArea.Strikethrough = true;
//Modifies the legend entry
chart.Legend.LegendEntries[0].IsDeleted = true;
//Modifies the legend layout - height, left, top, width
chart.Legend.Layout.Height = 50;
chart.Legend.Layout.HeightMode = LayoutModes.factor;
chart.Legend.Layout.Left = 10;
chart.Legend.Layout.LeftMode = LayoutModes.factor;
chart.Legend.Layout.Top = 30;
chart.Legend.Layout.TopMode = LayoutModes.factor;
chart.Legend.Layout.Width = 100;
chart.Legend.Layout.WidthMode = LayoutModes.factor;
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.docx);
//Closes the Word document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new
WordDocument((assembly.GetManifestResourceStream("XamarinFormsApp1.Assets.He
llo World.docx")),
FormatType.Docx)
{
//Gets the paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity
WChart chart = paragraph.ChildEntities[0] as WChart;
//Sets border settings - line color, pattern, weight, transparency
chart.PlotArea.Border.AutoFormat = false;
chart.PlotArea.Border.IsAutoLineColor = false;
}

```

```

chart.PlotArea.Border.LineColor = Syncfusion.Drawing.Color.Blue;
chart.PlotArea.Border.LinePattern = OfficeChartLinePattern.DashDot;
chart.PlotArea.Border.LineWeight = OfficeChartLineWeight.Wide;
chart.PlotArea.Border.Transparency = 0.6;
//Sets the plot area's fill type, color
chart.PlotArea.Fill.FillType = OfficeFillType.SolidColor;
chart.PlotArea.Fill.ForeColor = Syncfusion.Drawing.Color.LightPink;
//Sets the plot area shadow presence
chart.PlotArea.Shadow.ShadowInnerPresets =
Office2007ChartPresetsInner.InsideDiagonalTopLeft;
//Sets the legend position
chart.Legend.Position = OfficeLegendPosition.Left;
//Sets the layout inclusion
chart.Legend.IncludeInLayout = true;
//Sets the legend border format - color, pattern, weight
chart.Legend.FrameFormat.Border.AutoFormat = false;
chart.Legend.FrameFormat.Border.IsAutoLineColor = false;
chart.Legend.FrameFormat.Border.LineColor = Syncfusion.Drawing.Color.Blue;
chart.Legend.FrameFormat.Border.LinePattern =
OfficeChartLinePattern.DashDot;
chart.Legend.FrameFormat.Border.LineWeight = OfficeChartLineWeight.Wide;
//Sets the legend's text area formatting - font name, weight, color, size
chart.Legend.TextArea.Bold = true;
chart.Legend.TextArea.Color = OfficeKnownColors.Bright_green;
chart.Legend.TextArea.FontName = "Times New Roman";
chart.Legend.TextArea.Size = 20;
chart.Legend.TextArea.Strikethrough = true;
//Modifies the legend entry
chart.Legend.LegendEntries[0].IsDeleted = true;
//Modifies the legend layout - height, left, top, width
chart.Legend.Layout.Height = 50;
chart.Legend.Layout.HeightMode = LayoutModes.factor;
chart.Legend.Layout.Left = 10;
chart.Legend.Layout.LeftMode = LayoutModes.factor;
chart.Legend.Layout.Top = 30;
chart.Legend.Layout.TopMode = LayoutModes.factor;
chart.Legend.Layout.Width = 100;
chart.Legend.Layout.WidthMode = LayoutModes.factor;
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
//Closes the Word document
document.Close();
}

```

Positioning Chart Elements

The following code example illustrates how to specify the position of the chart elements.

C#

```

//Creates a new word document
WordDocument document = new WordDocument();
//Adds section to the document
IWSection sec = document.AddSection();
//Adds paragraph to the section
IWParagraph paragraph = sec.AddParagraph();
//Creates and Appends chart to the paragraph
WChart chart = paragraph.AppendChart(470, 300);
//Inputs data for chart
List<BarChartData> dataList = new List<BarChartData>();
BarChartData column1 = new BarChartData("P1", 286, 1300);
BarChartData column2 = new BarChartData("P2", 680, 700);
BarChartData column3 = new BarChartData("P3", 288, 1280);
BarChartData column4 = new BarChartData("P4", 200, 1200);
BarChartData column5 = new BarChartData("P5", 731, 2660);
dataList.Add(column1);
dataList.Add(column2);
dataList.Add(column3);
dataList.Add(column4);
dataList.Add(column5);
//Sets chart data by using IEnumerable overload
chart.SetDataRange(dataList, 1, 1);
//Sets chart type and title
chart.ChartTitle = "Purchase Details";
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Axis titles
chart.PrimaryCategoryAxis.Title = "Products";
chart.PrimaryValueAxis.Title = "In Dollars";
//Sets position for plot area
chart.PlotArea.Layout.LeftMode = LayoutModes.auto;
chart.PlotArea.Layout.TopMode = LayoutModes.factor;
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.outer;
//Sets position for title area
chart.ChartTitleArea.Layout.Left = 10;
chart.ChartTitleArea.Layout.Top = 8;
//Sets position for chart legend
chart.Legend.Layout.LeftMode = LayoutModes.factor;
chart.Legend.Layout.TopMode = LayoutModes.edge;
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new word document
Dim document As New WordDocument()
'Adds section to the document
Dim sec As IWSection = document.AddSection()
'Adds paragraph to the section
Dim paragraph As IWParagraph = sec.AddParagraph()
'Creates and Appends chart to the paragraph
Dim chart As WChart = paragraph.AppendChart(470, 300)
'Inputs data for chart
Dim dataList As New List(Of BarChartData)()
Dim column1 As New BarChartData("P1", 286, 1300)
Dim column2 As New BarChartData("P2", 680, 700)

```

```

Dim column3 As New BarChartData("P3", 288, 1280)
Dim column4 As New BarChartData("P4", 200, 1200)
Dim column5 As New BarChartData("P5", 731, 2660)
dataList.Add(column1)
dataList.Add(column2)
dataList.Add(column3)
dataList.Add(column4)
dataList.Add(column5)
'Sets chart data by using IEnumerable overload
chart.SetDataRange(dataList, 1, 1)
'Sets chart type and title
chart.ChartTitle = "Purchase Details"
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None
'Axis titles
chart.PrimaryCategoryAxis.Title = "Products"
chart.PrimaryValueAxis.Title = "In Dollars"
'Sets position for plot area
chart.PlotArea.Layout.LeftMode = LayoutModes.auto
chart.PlotArea.Layout.TopMode = LayoutModes.factor
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.outer
'Sets position for title area
chart.ChartTitleArea.Layout.Left = 10
chart.ChartTitleArea.Layout.Top = 8
'Sets position for chart legend
chart.Legend.Layout.LeftMode = LayoutModes.factor
chart.Legend.Layout.TopMode = LayoutModes.edge
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new word document
WordDocument document = new WordDocument();
//Adds section to the document
IWSection sec = document.AddSection();
//Adds paragraph to the section
IWParagraph paragraph = sec.AddParagraph();
//Creates and Appends chart to the paragraph
WChart chart = paragraph.AppendChart(470, 300);
//Inputs data for chart
List<BarChartData> dataList = new List<BarChartData>();
BarChartData column1 = new BarChartData("P1", 286, 1300);
BarChartData column2 = new BarChartData("P2", 680, 700);
BarChartData column3 = new BarChartData("P3", 288, 1280);
BarChartData column4 = new BarChartData("P4", 200, 1200);
BarChartData column5 = new BarChartData("P5", 731, 2660);
dataList.Add(column1);
dataList.Add(column2);
dataList.Add(column3);
dataList.Add(column4);
dataList.Add(column5);
//Sets chart data by using IEnumerable overload
chart.SetDataRange(dataList, 1, 1);
//Sets chart type and title
chart.ChartTitle = "Purchase Details";

```

```

chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Axis titles
chart.PrimaryCategoryAxis.Title = "Products";
chart.PrimaryValueAxis.Title = "In Dollars";
//Sets position for plot area
chart.PlotArea.Layout.LeftMode = LayoutModes.auto;
chart.PlotArea.Layout.TopMode = LayoutModes.factor;
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.outer;
//Sets position for title area
chart.ChartTitleArea.Layout.Left = 10;
chart.ChartTitleArea.Layout.Top = 8;
//Sets position for chart legend
chart.Legend.Layout.LeftMode = LayoutModes.factor;
chart.Legend.Layout.TopMode = LayoutModes.edge;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new word document
WordDocument document = new WordDocument();
//Adds section to the document
IWSection sec = document.AddSection();
//Adds paragraph to the section
IWParagraph paragraph = sec.AddParagraph();
//Creates and Appends chart to the paragraph
WChart chart = paragraph.AppendChart(470, 300);
//Inputs data for chart
List<BarChartData> dataList = new List<BarChartData>();
BarChartData column1 = new BarChartData("P1", 286, 1300);
BarChartData column2 = new BarChartData("P2", 680, 700);
BarChartData column3 = new BarChartData("P3", 288, 1280);
BarChartData column4 = new BarChartData("P4", 200, 1200);
BarChartData column5 = new BarChartData("P5", 731, 2660);
dataList.Add(column1);
dataList.Add(column2);
dataList.Add(column3);
dataList.Add(column4);
dataList.Add(column5);
//Sets chart data by using IEnumerable overload
chart.SetDataRange(dataList, 1, 1);
//Sets chart type and title
chart.ChartTitle = "Purchase Details";
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Axis titles
chart.PrimaryCategoryAxis.Title = "Products";
chart.PrimaryValueAxis.Title = "In Dollars";
//Sets position for plot area
chart.PlotArea.Layout.LeftMode = LayoutModes.auto;

```

```

chart.PlotArea.Layout.TopMode = LayoutModes.factor;
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.outer;
//Sets position for title area
chart.ChartTitleArea.Layout.Left = 10;
chart.ChartTitleArea.Layout.Top = 8;
//Sets position for chart legend
chart.Legend.Layout.LeftMode = LayoutModes.factor;
chart.Legend.Layout.TopMode = LayoutModes.edge;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
stream.Position = 0;
//Closes the document
document.Close();
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Creates a new word document
WordDocument document = new WordDocument();
//Adds section to the document
IWSection sec = document.AddSection();
//Adds paragraph to the section
IWParagraph paragraph = sec.AddParagraph();
//Creates and Appends chart to the paragraph
WChart chart = paragraph.AppendChart(470, 300);
//Inputs data for chart
List<BarChartData> dataList = new List<BarChartData>();
BarChartData column1 = new BarChartData("P1", 286, 1300);
BarChartData column2 = new BarChartData("P2", 680, 700);
BarChartData column3 = new BarChartData("P3", 288, 1280);
BarChartData column4 = new BarChartData("P4", 200, 1200);
BarChartData column5 = new BarChartData("P5", 731, 2660);
dataList.Add(column1);
dataList.Add(column2);
dataList.Add(column3);
dataList.Add(column4);
dataList.Add(column5);
//Sets chart data by using IEnumerable overload
chart.SetDataRange(dataList, 1, 1);
//Sets chart type and title
chart.ChartTitle = "Purchase Details";
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Axis titles
chart.PrimaryCategoryAxis.Title = "Products";
chart.PrimaryValueAxis.Title = "In Dollars";
//Sets position for plot area
chart.PlotArea.Layout.LeftMode = LayoutModes.auto;
chart.PlotArea.Layout.TopMode = LayoutModes.factor;
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.outer;
//Sets position for title area
chart.ChartTitleArea.Layout.Left = 10;
chart.ChartTitleArea.Layout.Top = 8;
//Sets position for chart legend
chart.Legend.Layout.LeftMode = LayoutModes.factor;

```

```
chart.Legend.Layout.TopMode = LayoutModes.edge;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
//Closes the document
document.Close();
```

The following code example describes the BarChartData class.

C#

```
public class BarChartData
{
    string name;
    int purchase;
    int expense;
    public string Name
    {
        get
        {
            return name;
        }
        set
        {
            name = value;
        }
    }
    public int Purchase
    {
        get
        {
            return purchase;
        }
        set
        {
            purchase = value;
        }
    }
    public int Expense
    {
        get
        {
            return expense;
        }
        set
        {
            expense = value;
        }
    }
}
```

```
public BarChartData(string name, int purchase, int expense)
{
    Name = name;
    Purchase = purchase;
    Expense = expense;
}
```

VB.NET

```
Public Class BarChartData
    Private m_name As String
    Private m_purchase As Integer
    Private m_expense As Integer
    Public Property Name() As String
    Get
        Return m_name
    End Get
    Set(value As String)
        m_name = value
    End Set
    End Property
    Public Property Purchase() As Integer
    Get
        Return m_purchase
    End Get
    Set(value As Integer)
        m_purchase = value
    End Set
    End Property
    Public Property Expense() As Integer
    Get
        Return m_expense
    End Get
    Set(value As Integer)
        m_expense = value
    End Set
    End Property
    Public Sub New(name__1 As String, purchase__2 As Integer, expense__3 As Integer)
        Name = name__1
        Purchase = purchase__2
        Expense = expense__3
    End Sub
End Class
```

UWP

```
public class BarChartData
{
    string name;
    int purchase;
    int expense;
    public string Name
    {
        get
```



```
{
    return name;
}
set
{
    name = value;
}
}
public int Purchase
{
    get
    {
        return purchase;
    }
    set
    {
        purchase = value;
    }
}
public int Expense
{
    get
    {
        return expense;
    }
    set
    {
        expense = value;
    }
}
public BarChartData(string name, int purchase, int expense)
{
    Name = name;
    Purchase = purchase;
    Expense = expense;
}
}
```

ASP.NET CORE

```
public class BarChartData
{
    string name;
    int purchase;
    int expense;
    public string Name
    {
        get
        {
            return name;
        }
        set
        {
            name = value;
        }
    }
}
```

```
public int Purchase
{
    get
    {
        return purchase;
    }
    set
    {
        purchase = value;
    }
}
public int Expense
{
    get
    {
        return expense;
    }
    set
    {
        expense = value;
    }
}
public BarChartData(string name, int purchase, int expense)
{
    Name = name;
    Purchase = purchase;
    Expense = expense;
}
```

XAMARIN

```
public class BarChartData
{
    string name;
    int purchase;
    int expense;
    public string Name
    {
        get
        {
            return name;
        }
        set
        {
            name = value;
        }
    }
    public int Purchase
    {
        get
        {
            return purchase;
        }
        set
        {
```

```

purchase = value;
}
}
public int Expense
{
    get
    {
        return expense;
    }
    set
    {
        expense = value;
    }
}
public BarChartData(string name, int purchase, int expense)
{
    Name = name;
    Purchase = purchase;
    Expense = expense;
}
}

```

Applying 3D Formats

Essential DocIO allows to modify the side wall, back wall, floor of the 3D chart. The following code snippet illustrates how to apply properties for side wall, floor and back wall of a 3D chart.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds section to the document
IWSection sec = document.AddSection();
//Adds paragraph to the section
IWParagraph paragraph = sec.AddParagraph();
//Loads the excel file as stream
Stream excelStream = File.OpenRead("Excel_Template.xlsx");
//Creates and Appends chart to the paragraph with excel stream as parameter
WChart chart = paragraph.AppendChart(excelStream, 1, "B2:C6", 470, 300);
//Sets chart type and title
chart.ChartType = OfficeChartType.Column_Clustered_3D;
chart.ChartTitle = "PurchaseDetails";
chart.ChartTitleArea.FontName = "Calibri";
chart.ChartTitleArea.Size = 14;
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None;
//Sets name to chart series
chart.Series[0].Name = "Sum of Purchases";
chart.Series[1].Name = "Sum of Future Expenses";
chart.PrimaryCategoryAxis.Title = "Products";
chart.PrimaryValueAxis.Title = "In Dollars";
//Sets position of legend
chart.Legend.Position = OfficeLegendPosition.Bottom;
//Sets rotation and elevation values
chart.Rotation = 20;
chart.Elevation = 15;
//Sets side wall properties
chart.SideWall.Fill.FillType = OfficeFillType.SolidColor;

```

```

chart.SideWall.Fill.ForeColor = Color.White;
chart.SideWall.Fill.BackColor = Color.White;
chart.SideWall.Border.LineColor = System.Drawing.Color.Beige;
//Sets floor fill option.
chart.Floor.Fill.FillType = OfficeFillType.Pattern;
//Sets the floor pattern Type.
chart.Floor.Fill.Pattern = OfficeGradientPattern.Pat_Divot;
//Sets the floor fore and Back ground color.
chart.Floor.Fill.ForeColor = System.Drawing.Color.Blue;
chart.Floor.Fill.BackColor = System.Drawing.Color.White;
//Sets the floor thickness.
chart.Floor.Thickness = 3;
//Sets the back wall fill option.
chart.BackWall.Fill.FillType = OfficeFillType.Gradient;
//Sets the Texture Type.
chart.BackWall.Fill.GradientColorType = OfficeGradientColor.TwoColor;
chart.BackWall.Fill.GradientStyle = OfficeGradientStyle.Diagonal_Down;
chart.BackWall.Fill.ForeColor = Color.WhiteSmoke;
chart.BackWall.Fill.BackColor = Color.LightBlue;
//Sets the Border Line color.
chart.BackWall.Border.LineColor = System.Drawing.Color.Wheat;
//Sets the Picture Type.
chart.BackWall.PictureUnit = OfficeChartPictureType.stretch;
//Sets the back wall thickness.
chart.BackWall.Thickness = 10;
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds section to the document
Dim sec As IWSection = document.AddSection()
'Adds paragraph to the section
Dim paragraph As IWParagraph = sec.AddParagraph()
'Loads the excel file as stream
Dim excelStream As Stream = File.OpenRead("Excel_Template.xlsx")
'Creates and Appends chart to the paragraph with excel stream as parameter
Dim chart As WChart = paragraph.AppendChart(excelStream, 1, "B2:C6", 470,
300)
'Sets chart type and title
chart.ChartType = OfficeChartType.Column_Clustered_3D
chart.ChartTitle = "Purchase Details"
chart.ChartTitleArea.FontName = "Calibri"
chart.ChartTitleArea.Size = 14
chart.ChartArea.Border.LinePattern = OfficeChartLinePattern.None
'Sets name to chart series
chart.Series(0).Name = "Sum of Purchases"
chart.Series(1).Name = "Sum of Future Expenses"
chart.PrimaryCategoryAxis.Title = "Products"
chart.PrimaryValueAxis.Title = "In Dollars"
'Sets position of legend
chart.Legend.Position = OfficeLegendPosition.Bottom
'Sets rotation and elevation values

```

```

chart.Rotation = 20
chart.Elevation = 15
'Sets side wall properties
chart.SideWall.Fill.FillType = OfficeFillType.SolidColor
chart.SideWall.Fill.ForeColor = Color.White
chart.SideWall.Fill.BackColor = Color.White
chart.SideWall.Border.LineColor = System.Drawing.Color.Beige
'Sets floor fill option.
chart.Floor.Fill.FillType = OfficeFillType.Pattern
'Sets the floor pattern Type.
chart.Floor.Fill.Pattern = OfficeGradientPattern.Pat_Divot
'Sets the floor fore and Back ground color.
chart.Floor.Fill.ForeColor = System.Drawing.Color.Blue
chart.Floor.Fill.BackColor = System.Drawing.Color.White
'Sets the floor thickness.
chart.Floor.Thickness = 3
'Sets the back wall fill option.
chart.BackWall.Fill.FillType = OfficeFillType.Gradient
'Sets the Texture Type.
chart.BackWall.Fill.GradientColorType = OfficeGradientColor.TwoColor
chart.BackWall.Fill.GradientStyle = OfficeGradientStyle.Diagonal_Down
chart.BackWall.Fill.ForeColor = Color.WhiteSmoke
chart.BackWall.Fill.BackColor = Color.LightBlue
'Sets the Border Line color.
chart.BackWall.Border.LineColor = System.Drawing.Color.Wheat
'Sets the Picture Type.
chart.BackWall.PictureUnit = OfficeChartPictureType.stretch
'Sets the back wall thickness.
chart.BackWall.Thickness = 10
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

Removing Chart

The following code example illustrates how to remove the chart from the document.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Template.docx");
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity and remove it from paragraph
foreach (ParagraphItem item in paragraph.ChildEntities)
{
    if (item is WChart)
    {
        paragraph.ChildEntities.Remove(item);
        break;
    }
}
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx")
'Gets the last paragraph
Dim paragraph As WParagraph = document.LastParagraph
'Gets the chart entity and removes it from paragraph
For Each item As ParagraphItem In paragraph.ChildEntities
If TypeOf item Is WChart Then
paragraph.ChildEntities.Remove(item)
Exit For
End If
Next
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Template.docx"),
FormatType.Docx);
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity and remove it from paragraph
foreach (ParagraphItem item in paragraph.ChildEntities)
{
if (item is WChart)
{
paragraph.ChildEntities.Remove(item);
break;
}
}
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads the template document
FileStream fileStreamPath = new FileStream("Template.docx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity and remove it from paragraph
foreach (ParagraphItem item in paragraph.ChildEntities)

```

```

{
    if (item is WChart)
    {
        paragraph.ChildEntities.Remove(item);
        break;
    }
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
stream.Position = 0;
//Closes the document
document.Close();
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Loads the template document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Templat
e.docx"), FormatType.Docx);
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Gets the chart entity and remove it from paragraph
foreach (ParagraphItem item in paragraph.ChildEntities)
{
    if (item is WChart)
    {
        paragraph.ChildEntities.Remove(item);
        break;
    }
}
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin

```

Supported Chart Types

The following chart types are supported in DocIO.

- Area
- Area_3D
- Area_Stacked
- AreaStacked100

- *AreaStacked100_3D*
- *AreaStacked3D*
- *Bar_Clustered*
- *BarClustered3D*
- *Bar_Stacked*
- *BarStacked100*
- *BarStacked100_3D*
- *BarStacked3D*
- *Bubble*
- *Bubble_3D*
- *Column_3D*
- *Column_Clustered*
- *ColumnClustered3D*
- *Column_Stacked*
- *ColumnStacked100*
- *ColumnStacked100_3D*
- *ColumnStacked3D*
- *Combination_Chart*
- *ConeBarClustered*
- *ConeBarStacked*
- *ConeBarStacked_100*
- *Cone_Clustered*
- *ConeClustered3D*
- *Cone_Stacked*
- *ConeStacked100*
- *CylinderBarClustered*
- *CylinderBarStacked*
- *CylinderBarStacked_100*
- *Cylinder_Clustered*
- *CylinderClustered3D*
- *Cylinder_Stacked*
- *CylinderStacked100*
- *Doughnut*
- *Doughnut_Exploded*
- *Line*
- *Line_3D*
- *Line_Markers*
- *LineMarkersStacked*
- *LineMarkersStacked_100*
- *Line_Stacked*
- *LineStacked100*
- *Pie*
- *Pie_3D*
- *Pie_Bar*
- *Pie_Exploded*
- *PieExploded3D*
- *PieOfPie*
- *PyramidBarClustered*

- `PyramidBarStacked`
- `PyramidBarStacked_100`
- `Pyramid_Clustered`
- `PyramidClustered3D`
- `Pyramid_Stacked`
- `PyramidStacked100`
- `Radar`
- `Radar_Filled`
- `Radar_Markers`
- `Scatter_Line`
- `ScatterLineMarkers`
- `Scatter_Markers`
- `Scatter_SmoothedLine`
- `ScatterSmoothedLineMarkers`
- `Stock_HighLowClose`
- `Stock_OpenHighLowClose`
- `Stock_VolumeHighLowClose`
- `Stock_VolumeOpenHighLowClose`
- `Surface_3D`
- `Surface_Contour`
- `SurfaceNoColor3D`
- `SurfaceNoColorContour`

Security

You can encrypt a Word document with password to restrict unauthorized access. You can also control the types of changes you make to this document.

Encrypting with password

The following code example shows how to encrypt the Word document with password.

C#

```
//Opens an input Word document
WordDocument document = new WordDocument("Template.docx");
//Encrypts the Word document with a password
document.EncryptDocument("password");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Opens an input Word document
Dim document As New WordDocument("Template.docx")
'Encrypts the Word document with a password
document.EncryptDocument("password")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates an instance of a WordDocument
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new WordDocument();
document.Open(assembly.GetManifestResourceStream("Sample.Assets.Template.docx"), FormatType.Docx);
//Encrypts the Word document with a password
document.EncryptDocument("password");
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//DocIO supports encryption in Windows Forms, WPF, ASP.NET, UWP, and ASP.NET MVC platforms alone.
```

XAMARIN

```
//DocIO supports encryption in Windows Forms, WPF, ASP.NET, UWP, and ASP.NET MVC platforms alone.
```

Opening the encrypted Word document

The following code example shows how to open the encrypted Word document.

C#

```
//Opens an encrypted Word document
WordDocument document = new WordDocument("Template.docx", "password");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Opens an encrypted Word document
Dim document As New WordDocument("Template.docx", "password")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
using (WordDocument document = new WordDocument())
{
    //Loads or opens an existing Word document from stream
}
```

```

Stream inputStream =
assembly.GetManifestResourceStream("CreateWordSample.Assets.Test.doc");
//Loads or opens an existing Word document through Open method of
WordDocument class
document.Open(inputStream, FormatType.Automatic, "password");
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Doc);
//Saves the stream as Word file in local machine
Save(stream, "Result.doc");
document.Close();
}
//Saves the Word document
async void Save(MemoryStream streams, string filename)
{
streams.Position = 0;
StorageFile stFile;
if
(! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
{
FileSavePicker savePicker = new FileSavePicker();
savePicker.DefaultFileExtension = ".doc";
savePicker.SuggestedFileName = filename;
savePicker.FileTypeChoices.Add("Word Documents", new List<string>()
{ ".doc" });
stFile = await savePicker.PickSaveFileAsync();
}
else
{
StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
stFile = await local.CreateFileAsync(filename,
CreationCollisionOption.ReplaceExisting);
}
if (stFile != null)
{
using (IRandomAccessStream zipStream = await
stFile.OpenAsync(FileAccessMode.ReadWrite))
{
//Write compressed data from memory to file
using (Stream ostream = zipStream.AsStreamForWrite())
{
byte[] buffer = streams.ToArray();
ostream.Write(buffer, 0, buffer.Length);
ostream.Flush();
}
}
}
//Launch the saved Word file
await Windows.System.Launcher.LaunchFileAsync(stFile);
}

```

ASP.NET CORE

```

//DocIO supports encryption in Windows Forms, WPF, ASP.NET, UWP, and ASP.NET
MVC platforms alone.

```

XAMARIN

//DocIO supports encryption in Windows Forms, WPF, ASP.NET, UWP, and ASP.NET MVC platforms alone.

Remove encryption

You can open the encrypted Word document and remove the encryption from the document. The following code example shows how to remove the encryption from encrypted Word document.

C#

```
//Opens an encrypted Word document
WordDocument document = new WordDocument ("Template.docx", "password");
//Remove encryption in Word document
document.RemoveEncryption();
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Opens an encrypted Word document
Dim document As New WordDocument("Template.docx", "password")
'Remove encryption in Word document
document.RemoveEncryption()
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Create new Word document instance
using (WordDocument document = new WordDocument())
{
    //Loads or opens an existing Word document from stream
    Stream inputStream =
    assembly.GetManifestResourceStream("Sample.Assets.Template.docx");
    //Loads or opens an existing Word document using the Open method of
WordDocument class
    document.Open(inputStream, FormatType.Docx, "password");
    //Remove encryption in Word document
    document.RemoveEncryption();
    MemoryStream stream = new MemoryStream();
    //Saves the Word file to MemoryStream
    document.Save(stream, FormatType.Docx);
    //Closes the Word document instance
    document.Close();
    //Saves the stream as Word file in local machine
    Save(stream, "Sample.Docx");
}
//Refer to the following link to save Word document in UWP platform
```

```
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//DocIO supports encryption in Windows Forms, WPF, ASP.NET, UWP, and ASP.NET MVC platforms alone.
```

XAMARIN

```
//DocIO supports encryption in Windows Forms, WPF, ASP.NET, UWP, and ASP.NET MVC platforms alone.
```

Protecting Word document from editing

You can restrict a Word document from editing either by providing a password or without password.

The following are the types of protection:

1. **AllowOnlyComments**: You can add/modify only the comments in the Word document.
2. **AllowOnlyFormFields**: You can modify the form field values in the Word document.
3. **AllowOnlyRevisions**: You can accept or reject the revisions in the Word document.
4. **AllowOnlyReading**: You can only view the content in the Word document.
5. **NoProtection**: You can access/edit the Word document contents as normally.

The following code example shows how to restrict editing to modify only form fields in a Word document.

C#

```
//Opens a Word document
WordDocument document = new WordDocument(@"Template.docx");
//Sets the protection with password and it allows only to modify the form fields type
document.Protect(ProtectionType.AllowOnlyFormFields, "password");
//Saves the Word document
document.Save("Protection.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Opens a Word document
Dim document As New WordDocument("Template.docx")
'Sets the protection with password and it allows only to modify the form fields type
document.Protect(ProtectionType.AllowOnlyFormFields, "password")
'Saves the Word document
document.Save("Protection.docx", FormatType.Docx)
document.Close()
```

UWP

```
//"App" is the class of Portable project
```

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new
WordDocument((assembly.GetManifestResourceStream("CreateWordSample.Assets.Te
st.docx")), FormatType.Docx))
{
//Sets the protection with password and it allows only to modify the form
fields type
document.Protect(ProtectionType.AllowOnlyFormFields, "password");
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Protection.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
//Closes the Word document
document.Close();
}

```

ASP.NET CORE

```

FileStream fileStreamPath = new FileStream(@"Data/Template.docx",
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new WordDocument(fileStreamPath,
FormatType.Docx))
{
//Sets the protection with password and it allows only to modify the form
fields type
document.Protect(ProtectionType.AllowOnlyFormFields, "password");
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.docx);
//Closes the Word document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Protection.docx");
}

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing document from file system through constructor of
WordDocument class
using (WordDocument document = new
WordDocument((assembly.GetManifestResourceStream("XamarinFormsAppl.Assets.He
llo World.docx")), FormatType.Docx))
{
//Sets the protection with password and it allows only to modify the form
fields type
document.Protect(ProtectionType.AllowOnlyFormFields, "password");
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
}

```

```
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Protection.docx",
"application/msword", stream);
//Closes the Word document
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
}
```

Footnotes and endnotes

Footnotes and endnotes are separate text body contents used in documents to show the source of supplementary information that does not interrupt the normal body text of the Word document. Footnotes are typically located at the bottom of a page or beneath text being referenced, and endnotes are typically placed at the end of a document or at the end of a section. When document has been divided up into one or more sections, each section of a document can contain endnotes.

Both footnotes and endnotes consist of two parts:

- A note reference mark with numbering value in the body text to indicate that additional information is in a footnote or endnote at the end of the page or the end of the document or section.
- The footnote or endnote text body content.

Adding a Footnotes

The following code example shows how to insert the footnotes into the Word document.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Creates a section
IWSection section = document.AddSection();
//Adds a paragraph to a section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to paragraph
paragraph.AppendText("Working with footnotes");
//Formats the text
paragraph.ApplyStyle(BuiltinStyle.Heading1);
//Adds a paragraph to a section
paragraph = section.AddParagraph();
//Appends the footnotes
WFootnote footnote = (WFootnote)
paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Footnote);
//Sets the footnote character format
footnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
//Inserts the text into the paragraph
paragraph.AppendText("Sample content for footnotes").CharacterFormat.Bold =
true;
//Adds footnote text
paragraph = footnote.TextBody.AddParagraph();
```

```

paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Creates a section
Dim section As ISection = document.AddSection()
'Adds a paragraph to a section
Dim paragraph As IParagraph = section.AddParagraph()
'Appends the text to paragraph
paragraph.AppendText("Working with footnotes")
'Formats the text
paragraph.ApplyStyle(BuiltinStyle.Heading1)
'Adds a paragraph to a section
paragraph = section.AddParagraph()
'Appends the footnotes
Dim footnote As WFootnote =
DirectCast(paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Footnote),
WFootnote)
'Sets the footnote character format
footnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript
'Inserts the text into the paragraph
paragraph.AppendText("Sample content for footnotes").CharacterFormat.Bold =
True
'Adds footnote text
paragraph = footnote.TextBody.AddParagraph()
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

using (WordDocument document = new WordDocument())
{
    //Creates a section
    ISection section = document.AddSection();
    //Adds a paragraph to a section
    IParagraph paragraph = section.AddParagraph();
    //Appends the text to paragraph
    paragraph.AppendText("Working with footnotes");
    //Formats the text
    paragraph.ApplyStyle(BuiltinStyle.Heading1);
    //Adds a paragraph to a section
    paragraph = section.AddParagraph();
    //Appends the footnotes
    WFootnote footnote =
(WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Footnote);

```



```
//Sets the footnote character format
footnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
//Inserts the text into the paragraph
paragraph.AppendText("Sample content for footnotes").CharacterFormat.Bold =
true;
//Adds footnote text
paragraph = footnote.TextBody.AddParagraph();
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
}
```

ASP.NET CORE

```
//Creates a new instance of WordDocument (Empty Word Document)
using (WordDocument document = new WordDocument())
{
    //Creates a section
    IWSection section = document.AddSection();
    //Adds a paragraph to a section
    IWParagraph paragraph = section.AddParagraph();
    //Appends the text to paragraph
    paragraph.AppendText("Working with footnotes");
    //Formats the text
    paragraph.ApplyStyle(BuiltinStyle.Heading1);
    //Adds a paragraph to a section
    paragraph = section.AddParagraph();
    //Appends the footnotes
    WFootnote footnote =
    (WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Footnote);
    //Sets the footnote character format
    footnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
    //Inserts the text into the paragraph
    paragraph.AppendText("Sample content for footnotes").CharacterFormat.Bold =
true;
    //Adds footnote text
    paragraph = footnote.TextBody.AddParagraph();
    paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
    MemoryStream stream = new MemoryStream();
    //Saves the Word document to MemoryStream
    document.Save(stream, FormatType.Docx);
    document.Close();
    stream.Position = 0;
    //Download Word document in the browser
    return File(stream, "application/msword", "Result.docx");
}
```

```
}

```

XAMARIN

```
using (WordDocument document = new WordDocument())
{
    //Creates a section
    IWSection section = document.AddSection();
    //Adds a paragraph to a section
    IWParagraph paragraph = section.AddParagraph();
    //Appends the text to paragraph
    paragraph.AppendText("Working with footnotes");
    //Formats the text
    paragraph.ApplyStyle(BuiltinStyle.Heading1);
    //Adds a paragraph to a section
    paragraph = section.AddParagraph();
    //Appends the footnotes
    WFootnote footnote =
        (WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Footnote);
    //Sets the footnote character format
    footnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
    //Inserts the text into the paragraph
    paragraph.AppendText("Sample content for footnotes").CharacterFormat.Bold =
        true;
    //Adds footnote text
    paragraph = footnote.TextBody.AddParagraph();
    paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
        the AdventureWorks sample databases are based, is a large, multinational
        manufacturing company.");
    MemoryStream stream = new MemoryStream();
    document.Save(stream, FormatType.Docx);
    //Save the stream as a file in the device and invoke it for viewing
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWorddoc.doc
        x", "application/msword", stream);
    //Closes the documents
    document.Close();
    //Please download the helper files from the below link to save the stream as
    file and open the file for viewing in Xamarin platform
    //https://help.syncfusion.com/file-formats/docio/create-word-document-in-
    xamarin#helper-files-for-xamarin
}
```

Adding a Endnotes

The following code example shows how to insert the endnotes into the Word document.

C#

```
//Creates a new document
WordDocument document = new WordDocument();
//Creates a section
IWSection section = document.AddSection();
//Adds a paragraph to a section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to paragraph
paragraph.AppendText("Working with endnotes");
//Formats the text
```

```

paragraph.ApplyStyle(BuiltinStyle.Heading1);
//Adds a paragraph to a section
paragraph = section.AddParagraph();
//Appends the endnote
WFootnote endnote = (WFootnote)
paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Endnote);
//Sets the endnote character format
endnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
//Inserts the text into the paragraph
paragraph.AppendText("Sample content for endnotes").CharacterFormat.Bold =
true;
//Adds footnote text
paragraph = endnote.TextBody.AddParagraph();
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new document
Dim document As New WordDocument()
'Creates a section
Dim section As IWSection = document.AddSection()
'Adds a paragraph to a section
Dim paragraph As IWParagraph = section.AddParagraph()
'Appends the text to paragraph
paragraph.AppendText("Working with endnotes")
'Formats the text
paragraph.ApplyStyle(BuiltinStyle.Heading1)
'Adds a paragraph to a section
paragraph = section.AddParagraph()
'Appends the endnote
Dim endnote As WFootnote =
DirectCast(paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Endnote),
WFootnote)
'Sets the endnote character format
endnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript
'Inserts the text into the paragraph
paragraph.AppendText("Sample content for endnotes").CharacterFormat.Bold =
True
'Adds footnote text
paragraph = endnote.TextBody.AddParagraph()
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

using (WordDocument document = new WordDocument())
{

```

```

//Creates a section
IWSection section = document.AddSection();
//Adds a paragraph to a section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to paragraph
paragraph.AppendText("Working with endnotes");
//Formats the text
paragraph.ApplyStyle(BuiltinStyle.Heading1);
//Adds a paragraph to a section
paragraph = section.AddParagraph();
//Appends the endnotes
WFootnote endnote =
(WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Endnote);
//Sets the endnote character format
endnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
//Inserts the text into the paragraph
paragraph.AppendText("Sample content for endnotes").CharacterFormat.Bold =
true;
//Adds endnote text
paragraph = endnote.TextBody.AddParagraph();
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
}

```

ASP.NET CORE

```

//Creates a new instance of WordDocument (Empty Word Document)
using (WordDocument document = new WordDocument())
{
//Creates a section
IWSection section = document.AddSection();
//Adds a paragraph to a section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to paragraph
paragraph.AppendText("Working with endnotes");
//Formats the text
paragraph.ApplyStyle(BuiltinStyle.Heading1);
//Adds a paragraph to a section
paragraph = section.AddParagraph();
//Appends the endnotes
WFootnote endnote =
(WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Endnote);
//Sets the endnote character format
endnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
//Inserts the text into the paragraph

```

```

paragraph.AppendText("Sample content for endnotes").CharacterFormat.Bold =
true;
//Adds endnote text
paragraph = endnote.TextBody.AddParagraph();
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

using (WordDocument document = new WordDocument())
{
    //Creates a section
    IWSection section = document.AddSection();
    //Adds a paragraph to a section
    IWParagraph paragraph = section.AddParagraph();
    //Appends the text to paragraph
    paragraph.AppendText("Working with endnotes");
    //Formats the text
    paragraph.ApplyStyle(BuiltinStyle.Heading1);
    //Adds a paragraph to a section
    paragraph = section.AddParagraph();
    //Appends the endnotes
    WFootnote endnote =
    (WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Endnote);
    //Sets the endnote character format
    endnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
    //Inserts the text into the paragraph
    paragraph.AppendText("Sample content for endnotes").CharacterFormat.Bold =
true;
    //Adds endnote text
    paragraph = endnote.TextBody.AddParagraph();
    paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
    MemoryStream stream = new MemoryStream();
    document.Save(stream, FormatType.Docx);
    //Save the stream as a file in the device and invoke it for viewing
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWorddoc.doc
x", "application/msword", stream);
    //Closes the documents
    document.Close();
    //Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
}

```

Footnote and Endnote separators

Footnote/Endnote separator is used to separate the text body content and footnote/endnote by a small line.

A footnote/endnote continuation separator is used to indicate the footnote/endnote is carried over from the previous page by a line running across the top section.

A footnote/endnote continuation notice is used to indicate the footnote/endnote is continued to the next page by special character or word preserved at the bottom of the footer.

The following code example shows how to change the default footnote separator.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Creates a section
IWSection section = document.AddSection();
//Adds a paragraph to a section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to paragraph
paragraph.AppendText("Working with footnotes");
//Formats the text
paragraph.ApplyStyle(BuiltinStyle.Heading1);
//Adds a paragraph to a section
paragraph = section.AddParagraph();
//Appends the footnotes
WFootnote footnote =
(WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Footnote);
WTextBody separator = document.Footnotes.Separator;
//Replaces the default footnote separated by text
separator.Paragraphs[0].Text = "Footnote separator";
//Sets the footnote character format
footnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
//Inserts the text into the paragraph
paragraph.AppendText("Sample content for footnotes").CharacterFormat.Bold =
true;
//Adds footnote text
paragraph = footnote.TextBody.AddParagraph();
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Creates a section
Dim section As IWSection = document.AddSection()
'Adds a paragraph to a section
Dim paragraph As IWParagraph = section.AddParagraph()
'Appends the text to paragraph
paragraph.AppendText("Working with footnotes")
'Formats the text
```

```

paragraph.ApplyStyle(BuiltinStyle.Heading1)
'Adds a paragraph to a section
paragraph = section.AddParagraph()
'Appends the footnotes
Dim footnote As WFootnote =
DirectCast(paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Footnote),
WFootnote)
Dim separator As WTextBody = document.Footnotes.Separator
'Replaces the default footnote separated by text
separator.Paragraphs(0).Text = "Footnote separator"
'Sets the footnote character format
footnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript
'Inserts the text into the paragraph
paragraph.AppendText("Sample content for footnotes").CharacterFormat.Bold =
True
'Adds footnote text
paragraph = footnote.TextBody.AddParagraph()
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

using (WordDocument document = new WordDocument())
{
    //Creates a section
    IWSection section = document.AddSection();
    //Adds a paragraph to a section
    IWParagraph paragraph = section.AddParagraph();
    //Appends the text to paragraph
    paragraph.AppendText("Working with footnotes");
    //Formats the text
    paragraph.ApplyStyle(BuiltinStyle.Heading1);
    //Adds a paragraph to a section
    paragraph = section.AddParagraph();
    //Appends the footnotes
    WFootnote footnote =
(WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Footnote);
    WTextBody separator = document.Footnotes.Separator;
    //Replaces the default footnote separated by text
    separator.Paragraphs[0].Text = "Footnote separator";
    //Sets the footnote character format
    footnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
    //Inserts the text into the paragraph
    paragraph.AppendText("Sample content for endnotes").CharacterFormat.Bold =
true;
    //Adds footnote text
    paragraph = footnote.TextBody.AddParagraph();
    paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
    MemoryStream stream = new MemoryStream();
    //Saves the Word file to MemoryStream

```

```

await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
}

```

ASP.NET CORE

```

//Creates a new instance of WordDocument (Empty Word Document)
using (WordDocument document = new WordDocument())
{
    //Creates a section
    IWSection section = document.AddSection();
    //Adds a paragraph to a section
    IWParagraph paragraph = section.AddParagraph();
    //Appends the text to paragraph
    paragraph.AppendText("Working with footnotes");
    //Formats the text
    paragraph.ApplyStyle(BuiltinStyle.Heading1);
    //Adds a paragraph to a section
    paragraph = section.AddParagraph();
    //Appends the footnotes
    WFootnote footnote =
        (WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Footnote);
    WTextBody separator = document.Footnotes.Separator;
    //Replaces the default footnote separated by text
    separator.Paragraphs[0].Text = "Footnote separator";
    //Sets the footnote character format
    footnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
    //Inserts the text into the paragraph
    paragraph.AppendText("Sample content for endnotes").CharacterFormat.Bold =
        true;
    //Adds footnote text
    paragraph = footnote.TextBody.AddParagraph();
    paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
        the AdventureWorks sample databases are based, is a large, multinational
        manufacturing company.");
    MemoryStream stream = new MemoryStream();
    //Saves the Word document to MemoryStream
    document.Save(stream, FormatType.Docx);
    document.Close();
    stream.Position = 0;
    //Download Word document in the browser
    return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

using (WordDocument document = new WordDocument())
{
    //Creates a section
    IWSection section = document.AddSection();
    //Adds a paragraph to a section

```



```

IWParagraph paragraph = section.AddParagraph();
//Appends the text to paragraph
paragraph.AppendText("Working with footnotes");
//Formats the text
paragraph.ApplyStyle(BuiltinStyle.Heading1);
//Adds a paragraph to a section
paragraph = section.AddParagraph();
//Appends the footnotes
WFootnote footnote =
(WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Footnote);
WTextBody separator = document.Footnotes.Separator;
//Replaces the default footnote separated by text
separator.Paragraphs[0].Text = "Footnote separator";
//Sets the footnote character format
footnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
//Inserts the text into the paragraph
paragraph.AppendText("Sample content for endnotes").CharacterFormat.Bold =
true;
//Adds footnote text
paragraph = footnote.TextBody.AddParagraph();
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWorddoc.doc
x", "application/msword", stream);
//Closes the documents
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
}

```

The following code example shows how to change the default endnote separator.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Creates a section
IWSection section = document.AddSection();
//Adds a paragraph to a section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to paragraph
paragraph.AppendText("Working with footnotes");
//Formats the text
paragraph.ApplyStyle(BuiltinStyle.Heading1);
//Adds a paragraph to section
paragraph = section.AddParagraph();
//Appends the endnote
WFootnote endnote =
(WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Endnote);
WTextBody separator = document.Endnotes.Separator;

```

```

//Replaces the default foot note separate by text
separator.Paragraphs[0].Text = "Endnote separator";
//Sets the endnote character format
endnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
//Inserts the text into the paragraph
paragraph.AppendText("Sample content for endnotes").CharacterFormat.Bold =
true;
//Adds the footnote text
paragraph = endnote.TextBody.AddParagraph();
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Creates a section
Dim section As IWSection = document.AddSection()
'Adds a paragraph to a section
Dim paragraph As IWParagraph = section.AddParagraph()
'Appends the text to paragraph
paragraph.AppendText("Working with footnotes")
'Formats the text
paragraph.ApplyStyle(BuiltinStyle.Heading1)
'Adds a paragraph to section
paragraph = section.AddParagraph()
'Appends the endnote
Dim endnote As WFootnote =
DirectCast(paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Endnote),
WFootnote)
Dim separator As WTextBody = document.Endnotes.Separator
'Replaces the default footnote separated by text
separator.Paragraphs(0).Text = "Endnote separator"
'Sets the endnote character format
endnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript
'Inserts the text into the paragraph
paragraph.AppendText("Sample content for endnotes").CharacterFormat.Bold =
True
'Adds the footnote text
paragraph = endnote.TextBody.AddParagraph()
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.")
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

using (WordDocument document = new WordDocument())
{
    //Creates a section

```

```

IWSection section = document.AddSection();
//Adds a paragraph to a section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to paragraph
paragraph.AppendText("Working with footnotes");
//Formats the text
paragraph.ApplyStyle(BuiltinStyle.Heading1);
//Adds a paragraph to a section
paragraph = section.AddParagraph();
//Appends the endnotes
WFootnote endnote =
(WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Endnote);
WTextBody separator = document.Endnotes.Separator;
//Replaces the default endnote separated by text
separator.Paragraphs[0].Text = "Endnote separator";
//Sets the endnote character format
endnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
//Inserts the text into the paragraph
paragraph.AppendText("Sample content for endnotes").CharacterFormat.Bold =
true;
//Adds endnote text
paragraph = endnote.TextBody.AddParagraph();
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
}

```

ASP.NET CORE

```

//Creates a new instance of WordDocument (Empty Word Document)
using (WordDocument document = new WordDocument())
{
//Creates a section
IWSection section = document.AddSection();
//Adds a paragraph to a section
IWParagraph paragraph = section.AddParagraph();
//Appends the text to paragraph
paragraph.AppendText("Working with footnotes");
//Formats the text
paragraph.ApplyStyle(BuiltinStyle.Heading1);
//Adds a paragraph to a section
paragraph = section.AddParagraph();
//Appends the endnotes
WFootnote endnote =
(WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Endnote);
WTextBody separator = document.Endnotes.Separator;
//Replaces the default endnote separated by text

```

```

separator.Paragraphs[0].Text = "Endnote separator";
//Sets the endnote character format
endnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
//Inserts the text into the paragraph
paragraph.AppendText("Sample content for endnotes").CharacterFormat.Bold =
true;
//Adds endnote text
paragraph = endnote.TextBody.AddParagraph();
paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Result.docx");
}

```

XAMARIN

```

using (WordDocument document = new WordDocument())
{
    //Creates a section
    IWSection section = document.AddSection();
    //Adds a paragraph to a section
    IWParagraph paragraph = section.AddParagraph();
    //Appends the text to paragraph
    paragraph.AppendText("Working with footnotes");
    //Formats the text
    paragraph.ApplyStyle(BuiltinStyle.Heading1);
    //Adds a paragraph to a section
    paragraph = section.AddParagraph();
    //Appends the endnotes
    WFootnote endnote =
    (WFootnote)paragraph.AppendFootnote(Syncfusion.DocIO.FootnoteType.Endnote);
    WTextBody separator = document.Endnotes.Separator;
    //Replaces the default endnote separated by text
    separator.Paragraphs[0].Text = "Endnote separator";
    //Sets the endnote character format
    endnote.MarkerCharacterFormat.SubSuperScript = SubSuperScript.SuperScript;
    //Inserts the text into the paragraph
    paragraph.AppendText("Sample content for endnotes").CharacterFormat.Bold =
true;
    //Adds endnote text
    paragraph = endnote.TextBody.AddParagraph();
    paragraph.AppendText("AdventureWorks Cycles, the fictitious company on which
the AdventureWorks sample databases are based, is a large, multinational
manufacturing company.");
    MemoryStream stream = new MemoryStream();
    document.Save(stream, FormatType.Docx);
    //Save the stream as a file in the device and invoke it for viewing
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WorkingWorddoc.doc
x", "application/msword", stream);
    //Closes the documents
}

```

```
document.Close();
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
}
```

Working with Macros

Macro is a way to automate the tasks that you perform repeatedly. It is a saved sequence of commands or keyboard strokes that can be recalled with a single command or keyboard stroke.

The following link shows how to create a macro in the Word document.

<https://support.office.com/en-in/article/Create-or-run-a-macro-c6b99036-905c-49a6-818a-dfb98b7c3c9c>

The following code illustrates how to load and save a macro enabled document.

C#

```
//Loads the macro-enabled template
WordDocument document = new WordDocument("Template.dotm");
//Gets the table
DataTable table = GetDataTable();
//Executes Mail Merge with groups
document.MailMerge.ExecuteGroup(table);
//Saves and closes the document
document.Save("Sample.docm", FormatType.Word2013Docm);
document.Close();
```

VB.NET

```
'Loads the macro-enabled template
Dim document As New WordDocument("Template.dotm")
'Gets the table
Dim table As DataTable = GetDataTable()
'Executes Mail Merge with groups
document.MailMerge.ExecuteGroup(table)
'Saves and closes the document
document.Save("Sample.docm", FormatType.Word2013Docm)
document.Close()
```

UWP

```
//DocIO supports mail merge operation in Windows Forms, WPF, ASP.NET and
ASP.NET MVC platforms alone.
```

ASP.NET CORE

```
//DocIO supports mail merge operation in Windows Forms, WPF, ASP.NET and
ASP.NET MVC platforms alone.
```

XAMARIN

//DocIO supports mail merge operation in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms alone.

The following code example illustrates the method used to get the tables from data set.

C#

```
private DataTable GetDataTable()  
{  
    //List of syncfusion products name  
    string[] products = { "DocIO", "PDF", "XlsIO" };  
    //Adds new Tables to the data set  
    DataRow row;  
    DataTable table = new DataTable();  
    //Adds fields to the Products table  
    table.TableName = "Products";  
    table.Columns.Add("ProductName");  
    table.Columns.Add("Binary");  
    table.Columns.Add("Source");  
    //Inserts values to the tables  
    foreach (string product in products)  
    {  
        row = table.NewRow();  
        row["ProductName"] = string.Concat("Essential ", product);  
        row["Binary"] = "$895.00";  
        row["Source"] = "$1,295.00";  
        table.Rows.Add(row);  
    }  
    return table;  
}
```

VB.NET

```
Private Function GetDataTable() As DataTable  
    'List of syncfusion products name  
    Dim products As String() = {"DocIO", "PDF", "XlsIO"}  
    'Adds new Tables to the data set  
    Dim row As DataRow  
    Dim table As New DataTable()  
    'Adds fields to the Products table  
    table.TableName = "Products"  
    table.Columns.Add("ProductName")  
    table.Columns.Add("Binary")  
    table.Columns.Add("Source")  
    'Inserts values to the tables  
    For Each product As String In products  
        row = table.NewRow()  
        row("ProductName") = String.Concat("Essential ", product)  
        row("Binary") = "$895.00"  
        row("Source") = "$1,295.00"  
        table.Rows.Add(row)  
    Next  
    Return table  
End Function
```

UWP

```
//DocIO supports mail merge operation in Windows Forms, WPF, ASP.NET and
ASP.NET MVC platforms alone.
```

ASP.NET CORE

```
//DocIO supports mail merge operation in Windows Forms, WPF, ASP.NET and
ASP.NET MVC platforms alone.
```

XAMARIN

```
//DocIO supports mail merge operation in Windows Forms, WPF, ASP.NET and
ASP.NET MVC platforms alone.
```

The following code example illustrates how to remove the macros present in the document by using RemoveMacros method.

C#

```
//Loads the document with macros
WordDocument document = new WordDocument("Sample.docm");
//Checks whether the document has macros and then removes them
if (document.HasMacros)
document.RemoveMacros();
//Saves the document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Loads the document with macros
Dim document As New WordDocument("Sample.docm")
'Checks whether the document has macros and then removes them
If document.HasMacros Then
document.RemoveMacros()
End If
'Saves the document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
//Loads the document with macros
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Sample.docm"),
FormatType.Docx);
//Checks whether the document has macros and then removes them
if (document.HasMacros)
document.RemoveMacros();
//Saves the Word file to MemoryStream
```

```

MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Closes the document
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Loads the document with macros
FileStream fileStreamPath = new FileStream("Sample.docm", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Checks whether the document has macros and then removes them
if (document.HasMacros)
document.RemoveMacros();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Loads the document with macros
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("GettingStarted.Data.Sample.
docm"), FormatType.Docx);
//Checks whether the document has macros and then removes them
if (document.HasMacros)
document.RemoveMacros();
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin

```


Working with Content Controls

What is Content Control?

Content controls are individual controls that you can add and customize to use in templates, forms, and documents. For example, many online forms are designed with a drop-down list control that provides a restricted set of choices.

Note: You can use content controls only in documents that are saved in the Open XML Format and cannot be used in the Word 97-2003 document (.doc) format.

Content controls can be categorized based on its occurrence in a document as follows,

- **InlineContentControl:** Among inline content inside, as a child of a paragraph.
- **BlockContentControl:** Among paragraphs and tables, as a child of a Body, HeaderFooter, Comment, Footnote, or a Shape node.

Block Content Control

You can add content control to a text body of the Word document using block content control. You can add text, tables, pictures, or other items into the block content control. Refer to the following code.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
WTextBody textBody = section.Body;
//Adds block content control into Word document
BlockContentControl blockContentControl =
textBody.AddBlockContentControl(ContentControlType.RichText) as
BlockContentControl;
//Adds new paragraph in the block content control
WParagraph paragraph = blockContentControl.TextBody.AddParagraph() as
WParagraph;
//Adds new text to the paragraph
paragraph.AppendText("Block content control");
//Adds new table to the block content control
WTable table = blockContentControl.TextBody.AddTable() as WTable;
//Specifies the total number of rows and columns
table.ResetCells(2, 3);
//Adds new paragraph to the block content control
paragraph = blockContentControl.TextBody.AddParagraph() as WParagraph;
//Adds image to the paragraph
paragraph.AppendPicture(Image.FromFile("Image.png"));
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds new section to the document
Dim section As IWSection = document.AddSection
Dim textBody As WTextBody = section.Body
```

```

'Adds block content control into Word document
Dim blockContentControl As BlockContentControl =
CType(textBody.AddBlockContentControl(ContentControlType.RichText),
BlockContentControl)
'Adds new paragraph in the block content control
Dim paragraph As WParagraph =
CType(blockContentControl.TextBody.AddParagraph, WParagraph)
'Adds new text to the paragraph
paragraph.AppendText("Block content control")
'Adds new table to the block content control
Dim table As WTable = CType(blockContentControl.TextBody.AddTable, WTable)
'Specifies the total number of rows and columns
table.ResetCells(2, 3)
'Adds new paragraph to the block content control
paragraph = CType(blockContentControl.TextBody.AddParagraph, WParagraph)
'Adds image to the paragraph
paragraph.AppendPicture(Image.FromFile("Image.png"))
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
WTextBody textBody = section.Body;
//Adds block content control into Word document
BlockContentControl blockContentControl =
textBody.AddBlockContentControl(ContentControlType.RichText) as
BlockContentControl;
//Adds new paragraph in the block content control
WParagraph paragraph = blockContentControl.TextBody.AddParagraph() as
WParagraph;
//Adds new text to the paragraph
paragraph.AppendText("Block content control");
//Adds new table to the block content control
WTable table = blockContentControl.TextBody.AddTable() as WTable;
//Specifies the total number of rows and columns
table.ResetCells(2, 3);
//Adds new paragraph to the block content control
paragraph = blockContentControl.TextBody.AddParagraph() as WParagraph;
//Adds image to the paragraph
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream pictureStream =
assembly.GetManifestResourceStream("Sample.Assets.Image.png");
paragraph.AppendPicture(pictureStream);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
WTextBody textBody = section.Body;
//Adds block content control into Word document
BlockContentControl blockContentControl =
textBody.AddBlockContentControl(ContentControlType.RichText) as
BlockContentControl;
//Adds new paragraph in the block content control
WParagraph paragraph = blockContentControl.TextBody.AddParagraph() as
WParagraph;
//Adds new text to the paragraph
paragraph.AppendText("Block content control");
//Adds new table to the block content control
WTable table = blockContentControl.TextBody.AddTable() as WTable;
//Specifies the total number of rows and columns
table.ResetCells(2, 3);
//Adds new paragraph to the block content control
paragraph = blockContentControl.TextBody.AddParagraph() as WParagraph;
//Gets the image stream
FileStream imageStream = new FileStream(@"D:\Image.png", FileMode.Open,
FileAccess.Read);
//Adds image to the paragraph
paragraph.AppendPicture(imageStream);
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
document.Close();
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
WTextBody textBody = section.Body;
//Adds block content control into Word document
BlockContentControl blockContentControl =
textBody.AddBlockContentControl(ContentControlType.RichText) as
BlockContentControl;
//Adds new paragraph in the block content control
WParagraph paragraph = blockContentControl.TextBody.AddParagraph() as
WParagraph;
//Adds new text to the paragraph
paragraph.AppendText("Block content control");
//Adds new table to the block content control
WTable table = blockContentControl.TextBody.AddTable() as WTable;
//Specifies the total number of rows and columns
table.ResetCells(2, 3);
//Adds new paragraph to the block content control
paragraph = blockContentControl.TextBody.AddParagraph() as WParagraph;
```

```
//Adds image to the paragraph
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Image.png");
//Adds image from stream
paragraph.AppendPicture(imageStream);
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", outputStream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document.Close();
```

Inline Content Control

You can add content control as a child to a paragraph using the inline content control. You can add text, pictures, fields or other paragraph items into the inline content control. Refer to the following code.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends inline content control to the paragraph
InlineContentControl inlineContentControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Inline content control ";
//Adds new text to the inline content control
inlineContentControl.ParagraphItems.Add(textRange);
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Adds new paragraph to the section
Dim paragraph As WParagraph = document.LastParagraph
'Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ")
```

```
'Appends inline content control to the paragraph
Dim inlineContentControl As InlineContentControl =
CType(paragraph.AppendInlineContentControl(ContentControlType.RichText),
InlineContentControl)
Dim textRange As WTextRange = New WTextRange(document)
textRange.Text = "Inline content control "
'Adds new text to the inline content control
inlineContentControl.ParagraphItems.Add(textRange)
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends inline content control to the paragraph
InlineContentControl inlineContentControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Inline content control ";
//Adds new text to the inline content control
inlineContentControl.ParagraphItems.Add(textRange);
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Result.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends inline content control to the paragraph
InlineContentControl inlineContentControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Inline content control ";
```

```
//Adds new text to the inline content control
inlineContentControl.ParagraphItems.Add(textRange);
//Creates memory stream.
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
document.Close();
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends inline content control to the paragraph
InlineContentControl inlineContentControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Inline content control ";
//Adds new text to the inline content control
inlineContentControl.ParagraphItems.Add(textRange);
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", outputStream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document.Close();
```

Note: Currently, DocIO does not support RowContentControl and CellContentControl.

Common properties of Content Control

You can set formatting options for the content control in the Word document. The following are the common properties of a content control.

Title

The title of the content control.

Tag

The tag value to identify the content control.

Appearance

This property allows you to define the appearance of the content controls. The appearance can be any one of the following:

- **BoundingBox:** Displays the contents of content control within a box.
- **Tags:** Displays the contents of content control within tags.
- **Hidden:** Displays the contents of content control without any box or tags.

Color

Defines the color of the content control.

Temporary

This property defines whether to remove a content control from the Word document when you edit the contents of the content control.

Lock Contents

Locking the contents of the content control. It restricts to modify the contents of the content control.

Lock Content Control

It restricts to remove or delete the content control.

Example – Content Control Common properties

The following code sample illustrates the content control properties usage.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends rich text content control to the paragraph
IInlineContentControl contentControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Rich text content control.";
//Adds new text to the rich text content control
contentControl.ParagraphItems.Add(textRange);
//Sets tag appearance for the content control
contentControl.ContentControlProperties.Appearance =
ContentControlAppearance.Tags;
//Sets a tag property to identify the content control
contentControl.ContentControlProperties.Tag = "Rich Text";
//Sets a title for the content control
contentControl.ContentControlProperties.Title = "Text";
//Sets the color for the content control
contentControl.ContentControlProperties.Color = Color.Magenta;
//Gets the type of content control
ContentControlType controlType =
contentControl.ContentControlProperties.Type;
//Enables content control lock
contentControl.ContentControlProperties.LockContentControl = true;
//Protects the contents of content control
contentControl.ContentControlProperties.LockContents = true;
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
```

```
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Adds new paragraph to the section
Dim paragraph As WParagraph = document.LastParagraph
'Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ")
'Appends rich text content control to the paragraph
Dim contentControl As IInlineContentControl =
CType(paragraph.AppendInlineContentControl(ContentControlType.RichText),
InlineContentControl)
Dim textRange As WTextRange = New WTextRange(document)
textRange.Text = "Rich text content control."
'Adds new text to the rich text content control
contentControl.ParagraphItems.Add(textRange)
'Sets tag appearance for the content control
contentControl.ContentControlProperties.Appearance =
ContentControlAppearance.Tags
'Sets a tag property to identify the content control
contentControl.ContentControlProperties.Tag = "Rich Text"
'Sets a title for the content control
contentControl.ContentControlProperties.Title = "Text"
'Sets the color for the content control
contentControl.ContentControlProperties.Color = Color.Magenta
'Gets the type of content control
Dim controlType As ContentControlType =
contentControl.ContentControlProperties.Type
'Enables content control lock
contentControl.ContentControlProperties.LockContentControl = True
'Protects the contents of content control
contentControl.ContentControlProperties.LockContents = True
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends rich text content control to the paragraph
IInlineContentControl contentControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Rich text content control.";
```



```

//Adds new text to the rich text content control
contentControl.ParagraphItems.Add(textRange);
//Sets tag appearance for the content control
contentControl.ContentControlProperties.Appearance =
ContentControlAppearance.Tags;
//Sets a tag property to identify the content control
contentControl.ContentControlProperties.Tag = "Rich Text";
//Sets a title for the content control
contentControl.ContentControlProperties.Title = "Text";
//Sets the color for the content control
contentControl.ContentControlProperties.Color = Color.Magenta;
//Gets the type of content control
ContentControlType controlType =
contentControl.ContentControlProperties.Type;
//Enables content control lock
contentControl.ContentControlProperties.LockContentControl = true;
//Protects the contents of content control
contentControl.ContentControlProperties.LockContents = true;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends rich text content control to the paragraph
IInlineContentControl contentControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Rich text content control.";
//Adds new text to the rich text content control
contentControl.ParagraphItems.Add(textRange);
//Sets tag appearance for the content control
contentControl.ContentControlProperties.Appearance =
ContentControlAppearance.Tags;
//Sets a tag property to identify the content control
contentControl.ContentControlProperties.Tag = "Rich Text";
//Sets a title for the content control
contentControl.ContentControlProperties.Title = "Text";
//Sets the color for the content control
contentControl.ContentControlProperties.Color =
Syncfusion.Drawing.Color.Magenta;

```

```
//Gets the type of content control
ContentControlType controlType =
contentControl.ContentControlProperties.Type;
//Enables content control lock
contentControl.ContentControlProperties.LockContentControl = true;
//Protects the contents of content control
contentControl.ContentControlProperties.LockContents = true;
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
document.Close();
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends rich text content control to the paragraph
IInlineContentControl contentControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Rich text content control.";
//Adds new text to the rich text content control
contentControl.ParagraphItems.Add(textRange);
//Sets tag appearance for the content control
contentControl.ContentControlProperties.Appearance =
ContentControlAppearance.Tags;
//Sets a tag property to identify the content control
contentControl.ContentControlProperties.Tag = "Rich Text";
//Sets a title for the content control
contentControl.ContentControlProperties.Title = "Text";
//Sets the color for the content control
contentControl.ContentControlProperties.Color =
Syncfusion.Drawing.Color.Magenta;
//Gets the type of content control
ContentControlType controlType =
contentControl.ContentControlProperties.Type;
//Enables content control lock
contentControl.ContentControlProperties.LockContentControl = true;
//Protects the contents of content control
contentControl.ContentControlProperties.LockContents = true;
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", outputStream);
```

```
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document.Close();
```

Why Content Control?

The content controls have the following three major use cases:

- Protection
- Form Filling
- Data Binding with Content Controls (XML Mapping)

Protection

Content controls provides options to prevent users from editing or deleting parts of a Word document contents. This is useful if you have information in a Word document or template that you should be able to read but not edit, or if you want to be able to edit content controls but not delete them.

To protect contents inside a content control, you can use properties of the content control to prevent editing or deleting the content control:

- The **LockContents** property prevents from editing the contents of the content control.
- The **LockContentControl** property prevents from deleting the content control.

The following code sample shows how to protect the content control and its contents.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends rich text content control to the paragraph
IInlineContentControl contentControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Rich text content control.";
//Adds new text to the rich text content control
contentControl.ParagraphItems.Add(textRange);
//Sets tag appearance for the content control
contentControl.ContentControlProperties.Appearance =
ContentControlAppearance.Tags;
//Sets a tag property to identify the content control
contentControl.ContentControlProperties.Tag = "Rich Text Protected";
//Sets a title for the content control
contentControl.ContentControlProperties.Title = "Text Protected";
//Enables content control lock
contentControl.ContentControlProperties.LockContentControl = true;
```

```
//Protects the contents of content control
contentControl.ContentControlProperties.LockContents = true;
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Adds new paragraph to the section
Dim paragraph As WParagraph = document.LastParagraph
'Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ")
'Appends rich text content control to the paragraph
Dim contentControl As IInlineContentControl =
CType(paragraph.AppendInlineContentControl(ContentControlType.RichText),
InlineContentControl)
Dim textRange As WTextRange = New WTextRange(document)
textRange.Text = "Rich text content control."
'Adds new text to the rich text content control
contentControl.ParagraphItems.Add(textRange)
'Sets tag appearance for the content control
contentControl.ContentControlProperties.Appearance =
ContentControlAppearance.Tags
'Sets a tag property to identify the content control
contentControl.ContentControlProperties.Tag = "Rich Text Protected"
'Sets a title for the content control
contentControl.ContentControlProperties.Title = "Text Protected"
'Enables content control lock
contentControl.ContentControlProperties.LockContentControl = True
'Protects the contents of content control
contentControl.ContentControlProperties.LockContents = True
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends rich text content control to the paragraph
IInlineContentControl contentControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Rich text content control.";
//Adds new text to the rich text content control
```

```

contentControl.ParagraphItems.Add(textRange);
//Sets tag appearance for the content control
contentControl.ContentControlProperties.Appearance =
ContentControlAppearance.Tags;
//Sets a tag property to identify the content control
contentControl.ContentControlProperties.Tag = "Rich Text Protected";
//Sets a title for the content control
contentControl.ContentControlProperties.Title = "Text Protected";
//Enables content control lock
contentControl.ContentControlProperties.LockContentControl = true;
//Protects the contents of content control
contentControl.ContentControlProperties.LockContents = true;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends rich text content control to the paragraph
IInlineContentControl contentControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Rich text content control.";
//Adds new text to the rich text content control
contentControl.ParagraphItems.Add(textRange);
//Sets tag appearance for the content control
contentControl.ContentControlProperties.Appearance =
ContentControlAppearance.Tags;
//Sets a tag property to identify the content control
contentControl.ContentControlProperties.Tag = "Rich Text Protected";
//Sets a title for the content control
contentControl.ContentControlProperties.Title = "Text Protected";
//Enables content control lock
contentControl.ContentControlProperties.LockContentControl = true;
//Protects the contents of content control
contentControl.ContentControlProperties.LockContents = true;
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
document.Close();

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends rich text content control to the paragraph
IInlineContentControl contentControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Rich text content control.";
//Adds new text to the rich text content control
contentControl.ParagraphItems.Add(textRange);
//Sets tag appearance for the content control
contentControl.ContentControlProperties.Appearance =
ContentControlAppearance.Tags;
//Sets a tag property to identify the content control
contentControl.ContentControlProperties.Tag = "Rich Text Protected";
//Sets a title for the content control
contentControl.ContentControlProperties.Title = "Text Protected";
//Enables content control lock
contentControl.ContentControlProperties.LockContentControl = true;
//Protects the contents of content control
contentControl.ContentControlProperties.LockContents = true;
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", outputStream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document.Close();

```

Form Filling

Another major use case is to create the forms. You can design your own forms for various stages using the text box, check box, and list box. Refer to the following code example.

Form creation:

C#

```

//Create a new document
WordDocument document = new WordDocument();
//Adding a new section to the document
IWSection section = document.AddSection();
//Adding a new paragraph to the section

```

```

IWParagraph paragraph = section.AddParagraph();
#region Document formatting
//Sets background color for document
document.Background.Gradient.Color1 = System.Drawing.Color.FromArgb(232,
232, 232);
document.Background.Gradient.Color2 = System.Drawing.Color.FromArgb(255,
255, 255);
document.Background.Type = BackgroundType.Gradient;
document.Background.Gradient.ShadingStyle = GradientShadingStyle.Horizontal;
document.Background.Gradient.ShadingVariant =
GradientShadingVariant.ShadingDown;
//Sets page size for document
section.PageSetup.Margins.All = 30f;
section.PageSetup.PageSize = new System.Drawing.SizeF(600, 600f);
#endregion
#region Title Section
//Adds a new table to the section
IWTable table = section.Body.AddTable();
table.ResetCells(1, 2);
//Gets the table first row
WTableRow row = table.Rows[0];
row.Height = 25f;
//Adds a new paragraph to the cell
IWParagraph cellPara = row.Cells[0].AddParagraph();
//Appends a new picture
IWPicture pic =
cellPara.AppendPicture(System.Drawing.Image.FromFile(@"D:\image.jpg"));
pic.Height = 80;
pic.Width = 180;
//Adds a new paragraph to the next cell
cellPara = row.Cells[1].AddParagraph();
row.Cells[1].CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle;
row.Cells[1].CellFormat.BackColor = System.Drawing.Color.FromArgb(173, 215,
255);
//Appends the text
IWTextRange txt = cellPara.AppendText("Job Application Form");
cellPara.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
//Sets the formats
txt.CharacterFormat.Bold = true;
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 18f;
//Sets the width and border type
row.Cells[0].Width = 200;
row.Cells[1].Width = 300;
row.Cells[1].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline;
#endregion
//Adds a new paragraph
section.AddParagraph();
#region General Information
//Adds a new table
table = section.Body.AddTable();
table.ResetCells(2, 1);
row = table.Rows[0];
row.Height = 20;

```

```

row.Cells[0].Width = 500;
//Adds a new paragraph
cellPara = row.Cells[0].AddParagraph();
//Sets a border type, color, background, and vertical alignment for cell
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Thick;
row.Cells[0].CellFormat.Borders.Color = System.Drawing.Color.FromArgb(155,
205, 255);
row.Cells[0].CellFormat.BackColor = System.Drawing.Color.FromArgb(198, 227,
255);
row.Cells[0].CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle;
txt = cellPara.AppendText(" General Information");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.Bold = true;
txt.CharacterFormat.FontSize = 11f;
row = table.Rows[1];
cellPara = row.Cells[0].AddParagraph();
//Sets a width, border type, color and background for cell
row.Cells[0].Width = 500;
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline;
row.Cells[0].CellFormat.Borders.Color = System.Drawing.Color.FromArgb(155,
205, 255);
row.Cells[0].CellFormat.BackColor = System.Drawing.Color.FromArgb(222, 239,
255);
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n Full Name:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
InlineContentControl txtField =
cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n Birth Date:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Date) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Date";
//Sets the date display format
txtField.ContentControlProperties.DateDisplayFormat = "M/d/yyyy";
//Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n Address:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;

```



```

//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets multiline property to true to get the multiple line input of Address
txtField.ContentControlProperties.Multiline = true;
//Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Phone:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Email:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
cellPara.AppendText("\n");
#endregion
section.AddParagraph();
#region Educational Qualification
//Adds a new table to the section
table = section.Body.AddTable();
table.ResetCells(2, 1);
row = table.Rows[0];
row.Height = 20;
//Sets width, border type, color, background and vertical alignment for cell
row.Cells[0].Width = 500;
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Thick;
row.Cells[0].CellFormat.Borders.Color = System.Drawing.Color.FromArgb(155,
205, 255);
row.Cells[0].CellFormat.BackColor = System.Drawing.Color.FromArgb(198, 227,
255);
row.Cells[0].CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle;
cellPara = row.Cells[0].AddParagraph();
//Appends a text to paragraph of cell
txt = cellPara.AppendText(" Educational Qualification");
txt.CharacterFormat.FontName = "Arial";

```

```

txt.CharacterFormat.Bold = true;
txt.CharacterFormat.FontSize = 11f;
row = table.Rows[1];
//Sets width, border type, color, and background for cell
row.Cells[0].Width = 500;
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline;
row.Cells[0].CellFormat.Borders.Color = System.Drawing.Color.FromArgb(155,
205, 255);
row.Cells[0].CellFormat.BackColor = System.Drawing.Color.FromArgb(222, 239,
255);
cellPara = row.Cells[0].AddParagraph();
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n Type:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
InlineContentControl dropdown =
cellPara.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
ContentControlListItem item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Higher";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Vocational";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Universal";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
dropdown.ContentControlProperties.Title = "Drop-Down";
//Sets formatting options for text present inside a content control
dropdown.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue;
dropdown.BreakCharacterFormat.FontName = "Arial";
dropdown.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Institution:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
//Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Programming Languages:");

```

```

txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n\t C#:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 9f;
//Appends a new inline content control to enter the value
dropdown =
cellPara.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
textRange = new WTextRange(document);
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Perfect";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Good";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Excellent";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
//Sets formatting options for text present inside a content control
dropdown.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue;
dropdown.BreakCharacterFormat.FontName = "Arial";
dropdown.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n\t VB:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 9f;
//Appends a new inline content control to enter the value
dropdown =
cellPara.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
textRange = new WTextRange(document);
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Perfect";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Good";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();

```

```

item.DisplayText = "Excellent";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
dropdown.ContentControlProperties.Title = "Drop-Down";
//Sets formatting options for text present inside a content control
dropdown.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue;
dropdown.BreakCharacterFormat.FontName = "Arial";
dropdown.BreakCharacterFormat.FontSize = 11f;
#endregion
//Saves and closes the Word document instance
document.Save("Form_Template.docx");
document.Close();

```

VB.NET

```

'Create a new document
Dim document As WordDocument = New WordDocument()
'Adding a new section to the document
Dim section As IWSection = document.AddSection()
'Adding a new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
#Region "Document Formatting"
'Sets background color for document
document.Background.Gradient.Color1 = System.Drawing.Color.FromArgb(232,
232, 232)
document.Background.Gradient.Color2 = System.Drawing.Color.FromArgb(255,
255, 255)
document.Background.Type = BackgroundType.Gradient
document.Background.Gradient.ShadingStyle = GradientShadingStyle.Horizontal
document.Background.Gradient.ShadingVariant =
GradientShadingVariant.ShadingDown
'Sets page size for document
section.PageSetup.Margins.All = 30.0F
section.PageSetup.PageSize = New System.Drawing.SizeF(600, 600.0!)
#End Region
#Region "Title Section"
'Adds a new table to the section
Dim table As IWTable = section.Body.AddTable()
table.ResetCells(1, 2)
'Gets the table first row
Dim row As WTableRow = table.Rows(0)
row.Height = 25.0F
'Adds a New paragraph to the cell
Dim cellPara As IWParagraph = row.Cells(0).AddParagraph
'Appends a new picture
Dim pic As IWPicture =
cellPara.AppendPicture(System.Drawing.Image.FromFile("D:\image.jpg"))
pic.Height = 80
pic.Width = 180
'Adds a new paragraph to the next cell
cellPara = row.Cells(1).AddParagraph
row.Cells(1).CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle
row.Cells(1).CellFormat.BackColor = System.Drawing.Color.FromArgb(173, 215,
255)
'Appends the text

```

```

Dim txt As ITextRange = cellPara.AppendText("Job Application Form")
cellPara.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center
'Sets the formats
txt.CharacterFormat.Bold = True
txt.CharacterFormat.FontName = "Arial"
txt.CharacterFormat.FontSize = 18.0F
'Sets the width and border type
row.Cells(0).Width = 200
row.Cells(1).Width = 300
row.Cells(1).CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline
#End Region
'Adds a new paragraph
section.AddParagraph()
#Region "General Information"
'Adds a new table
table = section.AddTable()
table.ResetCells(2, 1)
row = table.Rows(0)
row.Height = 20
row.Cells(0).Width = 500
'Adds a new paragraph
cellPara = row.Cells(0).AddParagraph
'Sets a border type, color, background, and vertical alignment for cell
row.Cells(0).CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Thick
row.Cells(0).CellFormat.Borders.Color = System.Drawing.Color.FromArgb(155,
205, 255)
row.Cells(0).CellFormat.BackColor = System.Drawing.Color.FromArgb(198, 227,
255)
row.Cells(0).CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle
txt = cellPara.AppendText(" General Information")
txt.CharacterFormat.FontName = "Arial"
txt.CharacterFormat.Bold = True
txt.CharacterFormat.FontSize = 11.0F
row = table.Rows(1)
cellPara = row.Cells(0).AddParagraph
'Sets a width, border type, color and background for cell
row.Cells(0).Width = 500
row.Cells(0).CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline
row.Cells(0).CellFormat.Borders.Color = System.Drawing.Color.FromArgb(155,
205, 255)
row.Cells(0).CellFormat.BackColor = System.Drawing.Color.FromArgb(222, 239,
255)
'Appends a text to paragraph of cell
txt = cellPara.AppendText(vbLf + " Full Name:" + vbTab + vbTab + vbTab +
vbTab)
txt.CharacterFormat.FontName = "Arial"
txt.CharacterFormat.FontSize = 11.0F
'Appends a new inline content control to enter the value
Dim txtField As InlineContentControl =
CType(cellPara.AppendInlineContentControl(ContentControlType.Text),
InlineContentControl)
txtField.ContentControlProperties.Title = "Text"

```

```

'Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue
txtField.BreakCharacterFormat.FontName = "Arial"
txtField.BreakCharacterFormat.FontSize = 11.0F
'Appends a text to paragraph of cell
txt = cellPara.AppendText(vbLf + vbLf + " Birth Date:" + vbTab + vbTab +
vbTab + vbTab)
txt.CharacterFormat.FontName = "Arial"
txt.CharacterFormat.FontSize = 11.0F
'Appends a new inline content control to enter the value
txtField =
CType(cellPara.AppendInlineContentControl(ContentControlType.Date),
InlineContentControl)
txtField.ContentControlProperties.Title = "Date"
'Sets the date display format
txtField.ContentControlProperties.DateDisplayFormat = "M/d/yyyy"
'Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue
txtField.BreakCharacterFormat.FontName = "Arial"
txtField.BreakCharacterFormat.FontSize = 11.0F
'Appends a text to paragraph of cell
txt = cellPara.AppendText(vbLf + vbLf + " Address:" + vbTab + vbTab + vbTab
+ vbTab)
txt.CharacterFormat.FontName = "Arial"
txt.CharacterFormat.FontSize = 11.0F
'Appends a new inline content control to enter the value
txtField =
CType(cellPara.AppendInlineContentControl(ContentControlType.Text),
InlineContentControl)
txtField.ContentControlProperties.Title = "Text"
'Sets multiline property to true to get the multiple line input of Address
txtField.ContentControlProperties.Multiline = True
'Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue
txtField.BreakCharacterFormat.FontName = "Arial"
txtField.BreakCharacterFormat.FontSize = 11.0F
'Appends a text to paragraph of cell
txt = cellPara.AppendText(vbLf + vbLf + " Phone:" + vbTab + vbTab + vbTab +
vbTab)
txt.CharacterFormat.FontName = "Arial"
txt.CharacterFormat.FontSize = 11.0F
'Appends a new inline content control to enter the value
txtField =
CType(cellPara.AppendInlineContentControl(ContentControlType.Text),
InlineContentControl)
txtField.ContentControlProperties.Title = "Text"
'Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue
txtField.BreakCharacterFormat.FontName = "Arial"
txtField.BreakCharacterFormat.FontSize = 11.0F
'Appends a text to paragraph of cell
txt = cellPara.AppendText(vbLf + vbLf + " Email:" + vbTab + vbTab + vbTab +
vbTab + vbTab)
txt.CharacterFormat.FontName = "Arial"
txt.CharacterFormat.FontSize = 11.0F
'Appends a New inline content control to enter the value

```

```

txtField =
CType(cellPara.AppendInlineContentControl(ContentControlType.Text),
InlineContentControl)
txtField.ContentControlProperties.Title = "Text"
'Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue
txtField.BreakCharacterFormat.FontName = "Arial"
txtField.BreakCharacterFormat.FontSize = 11.0F
cellPara.AppendText(vbLf)
#End Region
section.AddParagraph()
#Region "Educational Qualification"
'Adds a new table to the section
table = section.Body.AddTable()
table.ResetCells(2, 1)
row = table.Rows(0)
row.Height = 20
'Sets width, border type, color, background and vertical alignment for cell
row.Cells(0).Width = 500
row.Cells(0).CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Thick
row.Cells(0).CellFormat.Borders.Color = System.Drawing.Color.FromArgb(155,
205, 255)
row.Cells(0).CellFormat.BackColor = System.Drawing.Color.FromArgb(198, 227,
255)
row.Cells(0).CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle
cellPara = row.Cells(0).AddParagraph
'Appends a text to paragraph of cell
txt = cellPara.AppendText(" Educational Qualification")
txt.CharacterFormat.FontName = "Arial"
txt.CharacterFormat.Bold = True
txt.CharacterFormat.FontSize = 11.0F
row = table.Rows(1)
'Sets width, border type, color, and background for cell
row.Cells(0).Width = 500
row.Cells(0).CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline
row.Cells(0).CellFormat.Borders.Color = System.Drawing.Color.FromArgb(155,
205, 255)
row.Cells(0).CellFormat.BackColor = System.Drawing.Color.FromArgb(222, 239,
255)
cellPara = row.Cells(0).AddParagraph
'Appends a text to paragraph of cell
txt = cellPara.AppendText(vbLf + " Type:" + vbTab + vbTab + vbTab + vbTab +
vbTab)
txt.CharacterFormat.FontName = "Arial"
txt.CharacterFormat.FontSize = 11.0F
'Appends a new inline content control to enter the value
Dim dropdown As InlineContentControl =
CType(cellPara.AppendInlineContentControl(ContentControlType.DropDownList),
InlineContentControl)
Dim textRange As WTextRange = New WTextRange(document)
textRange.Text = "Choose an item from drop down list"
dropdown.ParagraphItems.Add(textRange)
'Creates an item for dropdown list
Dim item As ContentControlListItem = New ContentControlListItem()

```

```

'Sets the text to be displayed as list item
item.DisplayText = "Higher"
'Sets the value to the list item
item.Value = "1"
'Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item)
item = New ContentControlListItem()
item.DisplayText = "Vocational"
item.Value = "2"
dropdown.ContentControlProperties.ContentControlListItems.Add(item)
item = New ContentControlListItem()
item.DisplayText = "Universal"
item.Value = "3"
dropdown.ContentControlProperties.ContentControlListItems.Add(item)
dropdown.ContentControlProperties.Title = "Drop-Down"
'Sets formatting options for text present inside a content control
dropdown.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue
dropdown.BreakCharacterFormat.FontName = "Arial"
dropdown.BreakCharacterFormat.FontSize = 11.0F
'Appends a text to paragraph of cell
txt = cellPara.AppendText(vbLf + vbLf + " Institution:" + vbTab + vbTab +
vbTab + vbTab)
txt.CharacterFormat.FontName = "Arial"
txt.CharacterFormat.FontSize = 11.0F
'Appends a new inline content control to enter the value
txtField =
CType(cellPara.AppendInlineContentControl(ContentControlType.Text),
InlineContentControl)
'Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue
txtField.BreakCharacterFormat.FontName = "Arial"
txtField.BreakCharacterFormat.FontSize = 11.0F
'Appends a text to paragraph of cell
txt = cellPara.AppendText(vbLf + vbLf + " Programming Languages:")
txt.CharacterFormat.FontName = "Arial"
txt.CharacterFormat.FontSize = 11.0F
'Appends a text to paragraph of cell
txt = cellPara.AppendText("" + vbLf + vbLf + vbTab + " C#:" + vbTab + vbTab
+ vbTab + vbTab)
txt.CharacterFormat.FontName = "Arial"
txt.CharacterFormat.FontSize = 9.0F
'Appends a New inline content control to enter the value
dropdown =
CType(cellPara.AppendInlineContentControl(ContentControlType.DropDownList),
InlineContentControl)
textRange = New WTextRange(document)
textRange.Text = "Choose an item from drop down list"
dropdown.ParagraphItems.Add(textRange)
'Creates an item for dropdown list
item = New ContentControlListItem()
'Sets the text to be displayed as list item
item.DisplayText = "Perfect"
'Sets the value to the list item
item.Value = "1"
'Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item)
item = New ContentControlListItem()

```



```

item.DisplayText = "Good"
item.Value = "2"
dropdown.ContentControlProperties.ContentControlListItems.Add(item)
item = New ContentControlListItem()
item.DisplayText = "Excellent"
item.Value = "3"
dropdown.ContentControlProperties.ContentControlListItems.Add(item)
'Sets formatting options for text present inside a content control
dropdown.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue
dropdown.BreakCharacterFormat.FontName = "Arial"
dropdown.BreakCharacterFormat.FontSize = 11.0F
'Appends a text to paragraph of cell
txt = cellPara.AppendText("" + vbLf + vbLf + vbTab& + " VB:" + vbTab + vbTab
+ vbTab& + vbTab)
txt.CharacterFormat.FontName = "Arial"
txt.CharacterFormat.FontSize = 9.0F
'Appends a new inline content control to enter the value
dropdown =
CType(cellPara.AppendInlineContentControl(ContentControlType.DropDownList),
InlineContentControl)
textRange = New WTextRange(document)
textRange.Text = "Choose an item from drop down list"
dropdown.ParagraphItems.Add(textRange)
'Creates an item for dropdown list
item = New ContentControlListItem
'Sets the text to be displayed as list item
item.DisplayText = "Perfect"
'Sets the value to the list item
item.Value = "1"
'Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item)
item = New ContentControlListItem()
item.DisplayText = "Good"
item.Value = "2"
dropdown.ContentControlProperties.ContentControlListItems.Add(item)
item = New ContentControlListItem()
item.DisplayText = "Excellent"
item.Value = "3"
dropdown.ContentControlProperties.ContentControlListItems.Add(item)
dropdown.ContentControlProperties.Title = "Drop-Down"
'Sets formatting options for text present inside a content control
dropdown.BreakCharacterFormat.TextColor = System.Drawing.Color.MidnightBlue
dropdown.BreakCharacterFormat.FontName = "Arial"
dropdown.BreakCharacterFormat.FontSize = 11.0F
#End Region
'Saves and closes the Word document instance
document.Save("Form_Template.docx")
document.Close()

```

UWP

```

//Create a new document
WordDocument document = new WordDocument();
//Adding a new section to the document
IWSection section = document.AddSection();
//Adding a new paragraph to the section

```

```

IWParagraph paragraph = section.AddParagraph();
#region Document formatting
//Sets background color for document
document.Background.Gradient.Color1 = Color.FromArgb(232, 232, 232);
document.Background.Gradient.Color2 = Color.FromArgb(255, 255, 255);
document.Background.Type = BackgroundType.Gradient;
document.Background.Gradient.ShadingStyle = GradientShadingStyle.Horizontal;
document.Background.Gradient.ShadingVariant =
GradientShadingVariant.ShadingDown;
//Sets page size for document
section.PageSetup.Margins.All = 30f;
section.PageSetup.PageSize = new SizeF(600, 600f);
#endregion
#region Title Section
//Adds a new table to the section
IWTable table = section.Body.AddTable();
table.ResetCells(1, 2);
//Gets the table first row
WTableRow row = table.Rows[0];
row.Height = 25f;
//Adds a new paragraph to the cell
IWParagraph cellPara = row.Cells[0].AddParagraph();
//Appends a new picture
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream pictureStream =
assembly.GetManifestResourceStream("Sample.Assets.image.jpg");
IWPicture pic = cellPara.AppendPicture(pictureStream);
pic.Height = 80;
pic.Width = 180;
//Adds a new paragraph to the next cell
cellPara = row.Cells[1].AddParagraph();
row.Cells[1].CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle;
row.Cells[1].CellFormat.BackColor = Color.FromArgb(173, 215, 255);
//Appends the text
IWTextRange txt = cellPara.AppendText("Job Application Form");
cellPara.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
//Sets the formats
txt.CharacterFormat.Bold = true;
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 18f;
//Sets the width and border type
row.Cells[0].Width = 200;
row.Cells[1].Width = 300;
row.Cells[1].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline;
#endregion
//Adds a new paragraph
section.AddParagraph();
#region General Information
//Adds a new table
table = section.Body.AddTable();
table.ResetCells(2, 1);
row = table.Rows[0];
row.Height = 20;
row.Cells[0].Width = 500;

```

```

//Adds a new paragraph
cellPara = row.Cells[0].AddParagraph();
//Sets a border type, color, background, and vertical alignment for cell
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Thick;
row.Cells[0].CellFormat.Borders.Color = Color.FromArgb(155, 205, 255);
row.Cells[0].CellFormat.BackColor = Color.FromArgb(198, 227, 255);
row.Cells[0].CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle;
txt = cellPara.AppendText(" General Information");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.Bold = true;
txt.CharacterFormat.FontSize = 11f;
row = table.Rows[1];
cellPara = row.Cells[0].AddParagraph();
//Sets a width, border type, color and background for cell
row.Cells[0].Width = 500;
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline;
row.Cells[0].CellFormat.Borders.Color = Color.FromArgb(155, 205, 255);
row.Cells[0].CellFormat.BackColor = Color.FromArgb(222, 239, 255);
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n Full Name:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
InlineContentControl txtField =
cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Birth Date:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Date) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Date";
//Sets the date display format
txtField.ContentControlProperties.DateDisplayFormat = "M/d/yyyy";
//Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Address:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets multiline property to true to get the multiple line input of Address

```

```

txtField.ContentControlProperties.Multiline = true;
//Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Phone:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Email:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
cellPara.AppendText("\n");
#endregion
section.AddParagraph();
#region Educational Qualification
//Adds a new table to the section
table = section.Body.AddTable();
table.ResetCells(2, 1);
row = table.Rows[0];
row.Height = 20;
//Sets width, border type, color, background and vertical alignment for cell
row.Cells[0].Width = 500;
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Thick;
row.Cells[0].CellFormat.Borders.Color = Color.FromArgb(155, 205, 255);
row.Cells[0].CellFormat.BackColor = Color.FromArgb(198, 227, 255);
row.Cells[0].CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle;
cellPara = row.Cells[0].AddParagraph();
//Appends a text to paragraph of cell
txt = cellPara.AppendText(" Educational Qualification");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.Bold = true;
txt.CharacterFormat.FontSize = 11f;
row = table.Rows[1];
//Sets width, border type, color, and background for cell
row.Cells[0].Width = 500;
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline;

```

```

row.Cells[0].CellFormat.Borders.Color = Color.FromArgb(155, 205, 255);
row.Cells[0].CellFormat.BackColor = Color.FromArgb(222, 239, 255);
cellPara = row.Cells[0].AddParagraph();
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n Type:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
InlineContentControl dropdown =
cellPara.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
ContentControlListItem item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Higher";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Vocational";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Universal";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
dropdown.ContentControlProperties.Title = "Drop-Down";
//Sets formatting options for text present inside a content control
dropdown.BreakCharacterFormat.TextColor = Color.MidnightBlue;
dropdown.BreakCharacterFormat.FontName = "Arial";
dropdown.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Institution:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
//Sets formatting options for text present inside a content control
txtField.BreakCharacterFormat.TextColor = Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Programming Languages:");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n\t C#:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 9f;
//Appends a new inline content control to enter the value

```

```

dropdown =
cellPara.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
textRange = new WTextRange(document);
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Perfect";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Good";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Excellent";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
//Sets formatting options for text present inside a content control
dropdown.BreakCharacterFormat.TextColor = Color.MidnightBlue;
dropdown.BreakCharacterFormat.FontName = "Arial";
dropdown.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n\t VB:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 9f;
//Appends a new inline content control to enter the value
dropdown =
cellPara.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
textRange = new WTextRange(document);
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Perfect";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Good";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Excellent";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
dropdown.ContentControlProperties.Title = "Drop-Down";
//Sets formatting options for text present inside a content control
dropdown.BreakCharacterFormat.TextColor = Color.MidnightBlue;
dropdown.BreakCharacterFormat.FontName = "Arial";

```

```
dropdown.BreakCharacterFormat.FontSize = 11f;
#endregion
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Create a new document
WordDocument document = new WordDocument();
//Adding a new section to the document
IWSection section = document.AddSection();
//Adding a new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
#region Document formatting
//Sets background color for document
document.Background.Gradient.Color1 = Syncfusion.Drawing.Color.FromArgb(232,
232, 232);
document.Background.Gradient.Color2 = Syncfusion.Drawing.Color.FromArgb(255,
255, 255);
document.Background.Type = BackgroundType.Gradient;
document.Background.Gradient.ShadingStyle = GradientShadingStyle.Horizontal;
document.Background.Gradient.ShadingVariant =
GradientShadingVariant.ShadingDown;
//Sets page size for document
section.PageSetup.Margins.All = 30f;
section.PageSetup.PageSize = new Syncfusion.Drawing.SizeF(600, 600f);
#endregion
#region Title Section
//Adds a new table to the section
IWTable table = section.Body.AddTable();
table.ResetCells(1, 2);
//Gets the table first row
WTableRow row = table.Rows[0];
row.Height = 25f;
//Adds a new paragraph to the cell
IWParagraph cellPara = row.Cells[0].AddParagraph();
//Appends new picture
IWPicture pic = cellPara.AppendPicture(new FileStream(@"D:\image.jpg",
 FileMode.Open, FileAccess.Read));
pic.Height = 80;
pic.Width = 180;
//Adds a new paragraph to the next cell
cellPara = row.Cells[1].AddParagraph();
row.Cells[1].CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle;
row.Cells[1].CellFormat.BackColor = Syncfusion.Drawing.Color.FromArgb(173,
215, 255);
//Appends the text
IWTextRange txt = cellPara.AppendText("Job Application Form");
```

```

cellPara.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
//Sets the formats
txt.CharacterFormat.Bold = true;
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 18f;
//Sets the widths and border types
row.Cells[0].Width = 200;
row.Cells[1].Width = 300;
row.Cells[1].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline;
#endregion
//Adds a new paragraph
section.AddParagraph();
#region General Information
//Adds a new table
table = section.Body.AddTable();
table.ResetCells(2, 1);
row = table.Rows[0];
row.Height = 20;
row.Cells[0].Width = 500;
//Adds a new paragraph
cellPara = row.Cells[0].AddParagraph();
//Sets a border type, color and background for cell
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Thick;
row.Cells[0].CellFormat.Borders.Color =
Syncfusion.Drawing.Color.FromArgb(155, 205, 255);
row.Cells[0].CellFormat.BackColor = Syncfusion.Drawing.Color.FromArgb(198,
227, 255);
row.Cells[0].CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle;
txt = cellPara.AppendText(" General Information");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.Bold = true;
txt.CharacterFormat.FontSize = 11f;
row = table.Rows[1];
cellPara = row.Cells[0].AddParagraph();
//Sets a width, border type, color and background for cell
row.Cells[0].Width = 500;
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline;
row.Cells[0].CellFormat.Borders.Color =
Syncfusion.Drawing.Color.FromArgb(155, 205, 255);
row.Cells[0].CellFormat.BackColor = Syncfusion.Drawing.Color.FromArgb(222,
239, 255);
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n Full Name:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
InlineContentControl txtField =
cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets formatting options for text present insider a content control

```



```

txtField.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Birth Date:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Date) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Date";
//Sets the date display format
txtField.ContentControlProperties.DateDisplayFormat = "M/d/yyyy";
//Sets formatting options for text present insider a content control
txtField.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Address:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets multiline property to true to get the multiple line input of Address
txtField.ContentControlProperties.Multiline = true;
//Sets formatting options for text present insider a content control
txtField.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Phone:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets formatting options for text present insider a content control
txtField.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Email:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets formatting options for text present insider a content control

```

```

txtField.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
cellPara.AppendText("\n");
#endregion
section.AddParagraph();
#region Educational Qualification
//Adds a new table to the section
table = section.Body.AddTable();
table.ResetCells(2, 1);
row = table.Rows[0];
row.Height = 20;
//Sets width, border type, color, background and vertical alignment for cell
row.Cells[0].Width = 500;
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Thick;
row.Cells[0].CellFormat.Borders.Color =
Syncfusion.Drawing.Color.FromArgb(155, 205, 255);
row.Cells[0].CellFormat.BackColor = Syncfusion.Drawing.Color.FromArgb(198,
227, 255);
row.Cells[0].CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle;
cellPara = row.Cells[0].AddParagraph();
//Appends a text to paragraph of cell
txt = cellPara.AppendText(" Educational Qualification");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.Bold = true;
txt.CharacterFormat.FontSize = 11f;
row = table.Rows[1];
//Sets width, border type, color, and background for cell
row.Cells[0].Width = 500;
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline;
row.Cells[0].CellFormat.Borders.Color =
Syncfusion.Drawing.Color.FromArgb(155, 205, 255);
row.Cells[0].CellFormat.BackColor = Syncfusion.Drawing.Color.FromArgb(222,
239, 255);
cellPara = row.Cells[0].AddParagraph();
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n Type:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
InlineContentControl dropdown =
cellPara.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
ContentControlListItem item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Higher";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list

```

```

dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Vocational";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Universal";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
dropdown.ContentControlProperties.Title = "Drop-Down";
//Sets formatting options for text present insider a content control
dropdown.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
dropdown.BreakCharacterFormat.FontName = "Arial";
dropdown.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Institution:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
//Sets formatting options for text present insider a content control
txtField.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Programming Languages:");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n\t C#:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 9f;
//Appends a new inline content control to enter the value
dropdown =
cellPara.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
textRange = new WTextRange(document);
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Perfect";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Good";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Excellent";
item.Value = "3";

```

```

dropdown.ContentControlProperties.ContentControlListItems.Add(item);
//Sets formatting options for text present insider a content control
dropdown.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
dropdown.BreakCharacterFormat.FontName = "Arial";
dropdown.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n\t VB:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 9f;
//Appends a new inline content control to enter the value
dropdown =
cellPara.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
textRange = new WTextRange(document);
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Perfect";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Good";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Excellent";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
dropdown.ContentControlProperties.Title = "Drop-Down";
//Sets formatting options for text present insider a content control
dropdown.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
dropdown.BreakCharacterFormat.FontName = "Arial";
dropdown.BreakCharacterFormat.FontSize = 11f;
#endregion
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
document.Close();

```

XAMARIN

```

//Create a new document
WordDocument document = new WordDocument();
//Adding a new section to the document
IWSection section = document.AddSection();
//Adding a new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
#region Document formatting
//Sets background color for document

```

```

document.Background.Gradient.Color1 = Syncfusion.Drawing.Color.FromArgb(232,
232, 232);
document.Background.Gradient.Color2 = Syncfusion.Drawing.Color.FromArgb(255,
255, 255);
document.Background.Type = BackgroundType.Gradient;
document.Background.Gradient.ShadingStyle = GradientShadingStyle.Horizontal;
document.Background.Gradient.ShadingVariant =
GradientShadingVariant.ShadingDown;
//Sets page size for document
section.PageSetup.Margins.All = 30f;
section.PageSetup.PageSize = new Syncfusion.Drawing.SizeF(600, 600f);
#endregion
#region Title Section
//Adds a new table to the section
IWTable table = section.Body.AddTable();
table.ResetCells(1, 2);
//Gets the table first row
WTableRow row = table.Rows[0];
row.Height = 25f;
//Adds a new paragraph to the cell
IWParagraph cellPara = row.Cells[0].AddParagraph();
//Appends new picture
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Image.jpg");
IWPicture pic = cellPara.AppendPicture(imageStream);
pic.Height = 80;
pic.Width = 180;
//Adds a new paragraph to the next cell
cellPara = row.Cells[1].AddParagraph();
row.Cells[1].CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle;
row.Cells[1].CellFormat.BackColor = Syncfusion.Drawing.Color.FromArgb(173,
215, 255);
//Appends the text
IWTextRange txt = cellPara.AppendText("Job Application Form");
cellPara.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
//Sets the formats
txt.CharacterFormat.Bold = true;
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 18f;
//Sets the width and border type
row.Cells[0].Width = 200;
row.Cells[1].Width = 300;
row.Cells[1].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline;
#endregion
//Adds a new paragraph
section.AddParagraph();
#region General Information
//Adds a new table
table = section.Body.AddTable();
table.ResetCells(2, 1);
row = table.Rows[0];
row.Height = 20;
row.Cells[0].Width = 500;

```

```

//Adds a new paragraph
cellPara = row.Cells[0].AddParagraph();
//Sets a border type, color, background, and vertical alignment for cell
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Thick;
row.Cells[0].CellFormat.Borders.Color =
Syncfusion.Drawing.Color.FromArgb(155, 205, 255);
row.Cells[0].CellFormat.BackColor = Syncfusion.Drawing.Color.FromArgb(198,
227, 255);
row.Cells[0].CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle;
txt = cellPara.AppendText(" General Information");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.Bold = true;
txt.CharacterFormat.FontSize = 11f;
row = table.Rows[1];
cellPara = row.Cells[0].AddParagraph();
//Sets a width, border type, color and background for cell
row.Cells[0].Width = 500;
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline;
row.Cells[0].CellFormat.Borders.Color =
Syncfusion.Drawing.Color.FromArgb(155, 205, 255);
row.Cells[0].CellFormat.BackColor = Syncfusion.Drawing.Color.FromArgb(222,
239, 255);
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n Full Name:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
InlineContentControl txtField =
cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets formatting options for text present insider a content control.
txtField.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Birth Date:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Date) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Date";
//Sets the date display format
txtField.ContentControlProperties.DateDisplayFormat = "M/d/yyyy";
//Sets formatting options for text present insider a content control
txtField.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Address:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";

```

```

txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets multiline property to true to get the multiple line input of Address
txtField.ContentControlProperties.Multiline = true;
//Sets formatting options for text present insider a content control
txtField.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Phone:\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets formatting options for text present insider a content control
txtField.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Email:\t\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
txtField.ContentControlProperties.Title = "Text";
//Sets formatting options for text present insider a content control
txtField.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
cellPara.AppendText("\n");
#endregion
section.AddParagraph();
#region Educational Qualification
//Adds a new table to the section
table = section.Body.AddTable();
table.ResetCells(2, 1);
row = table.Rows[0];
row.Height = 20;
//Sets width, border type, color, background and vertical alignment for cell
row.Cells[0].Width = 500;
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Thick;
row.Cells[0].CellFormat.Borders.Color =
Syncfusion.Drawing.Color.FromArgb(155, 205, 255);
row.Cells[0].CellFormat.BackColor = Syncfusion.Drawing.Color.FromArgb(198,
227, 255);
row.Cells[0].CellFormat.VerticalAlignment =
Syncfusion.DocIO.DLS.VerticalAlignment.Middle;

```

```

cellPara = row.Cells[0].AddParagraph();
//Appends a text to paragraph of cell
txt = cellPara.AppendText(" Educational Qualification");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.Bold = true;
txt.CharacterFormat.FontSize = 11f;
row = table.Rows[1];
//Sets width, border type, color, and background for cell
row.Cells[0].Width = 500;
row.Cells[0].CellFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Hairline;
row.Cells[0].CellFormat.Borders.Color =
Syncfusion.Drawing.Color.FromArgb(155, 205, 255);
row.Cells[0].CellFormat.BackColor = Syncfusion.Drawing.Color.FromArgb(222,
239, 255);
cellPara = row.Cells[0].AddParagraph();
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n Type:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
InlineContentControl dropdown =
cellPara.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
ContentControlListItem item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Higher";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Vocational";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Universal";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
dropdown.ContentControlProperties.Title = "Drop-Down";
//Sets formatting options for text present insider a content control
dropdown.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
dropdown.BreakCharacterFormat.FontName = "Arial";
dropdown.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Institution:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a new inline content control to enter the value
txtField = cellPara.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
//Sets formatting options for text present insider a content control

```



```

txtField.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
txtField.BreakCharacterFormat.FontName = "Arial";
txtField.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n Programming Languages:");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n\t C#:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 9f;
//Appends a new inline content control to enter the value
dropdown =
cellPara.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
textRange = new WTextRange(document);
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Perfect";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Good";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Excellent";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
//Sets formatting options for text present insider a content control
dropdown.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
dropdown.BreakCharacterFormat.FontName = "Arial";
dropdown.BreakCharacterFormat.FontSize = 11f;
//Appends a text to paragraph of cell
txt = cellPara.AppendText("\n\n\t VB:\t\t\t\t\t");
txt.CharacterFormat.FontName = "Arial";
txt.CharacterFormat.FontSize = 9f;
//Appends a new inline content control to enter the value
dropdown =
cellPara.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
textRange = new WTextRange(document);
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Perfect";
//Sets the value to the list item
item.Value = "1";

```

```

//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Good";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Excellent";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
dropdown.ContentControlProperties.Title = "Drop-Down";
//Sets formatting options for text present insider a content control
dropdown.BreakCharacterFormat.TextColor =
Syncfusion.Drawing.Color.MidnightBlue;
dropdown.BreakCharacterFormat.FontName = "Arial";
dropdown.BreakCharacterFormat.FontSize = 11f;
#endregion
//Create memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", outputStream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document.Close();

```

You can also fill the forms using the DocIO. Refer to the following code example.

Form filling:

C#

```

//Open the created form document.
WordDocument document1 = new WordDocument("Form_Template.docx");
IWSection sec = document1.LastSection;
InlineContentControl inlineCC;
InlineContentControl dropDownCC;
WTable table1 = sec.Tables[1] as WTable;
WTableRow row1 = table1.Rows[1];
#region General Information
//Fill the name
WParagraph cellPara1 = row1.Cells[0].ChildEntities[1] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
WTextRange text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "Steve Jobs";
inlineCC.ParagraphItems.Add(text);
//Fill the date of birth
cellPara1 = row1.Cells[0].ChildEntities[3] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "06/01/1994";

```

```

inlineCC.ParagraphItems.Add(text);
//Fill the address
cellPara1 = row1.Cells[0].ChildEntities[5] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "2501 Aerial Center Parkway.";
inlineCC.ParagraphItems.Add(text);
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "Morrisville, NC 27560.";
inlineCC.ParagraphItems.Add(text);
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "USA.";
inlineCC.ParagraphItems.Add(text);
//Fill the phone no
cellPara1 = row1.Cells[0].ChildEntities[7] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "+1 919.481.1974";
inlineCC.ParagraphItems.Add(text);
//Fill the email id
cellPara1 = row1.Cells[0].ChildEntities[9] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "steve123@email.com";
inlineCC.ParagraphItems.Add(text);
#endregion
#region Educational Information
table1 = sec.Tables[2] as WTable;
row1 = table1.Rows[1];
//Fill the education type
cellPara1 = row1.Cells[0].ChildEntities[1] as WParagraph;
dropDownCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems[1].DisplayText;
dropDownCC.ParagraphItems.Add(text);
//Fill the university
cellPara1 = row1.Cells[0].ChildEntities[3] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text = "Michigan University";
inlineCC.ParagraphItems.Add(text);
//Fill the C# experience level
cellPara1 = row1.Cells[0].ChildEntities[7] as WParagraph;
dropDownCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems[2].DisplayText;
dropDownCC.ParagraphItems.Add(text);

```

```

//Fill the VB experience level
cellPara1 = row1.Cells[0].ChildEntities[9] as WParagraph;
dropDownCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems[1].DisplayText;
dropDownCC.ParagraphItems.Add(text);
#endregion
//Saves and closes the Word document instance
document1.Save("Form_Filled.docx");
document1.Close();

```

VB.NET

```

'Open the created form document
Dim document1 As WordDocument = New WordDocument("Form_Template.docx")
Dim sec As IWSection = document1.LastSection
Dim inlineCC As InlineContentControl
Dim dropDownCC As InlineContentControl
Dim table1 As WTable = CType(sec.Tables(1), WTable)
Dim row1 As WTableRow = table1.Rows(1)
#Region "General Information"
Dim cellPara1 As WParagraph = CType(row1.Cells(0).ChildEntities(1),
WParagraph)
inlineCC = CType(cellPara1.ChildEntities.LastItem, InlineContentControl)
Dim text As WTextRange = New WTextRange(document1)
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat)
text.Text = "Steve Jobs"
inlineCC.ParagraphItems.Add(text)
'Fill the date of birth
cellPara1 = CType(row1.Cells(0).ChildEntities(3), WParagraph)
inlineCC = CType(cellPara1.ChildEntities.LastItem, InlineContentControl)
text = New WTextRange(document1)
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat)
text.Text = "06/01/1994"
inlineCC.ParagraphItems.Add(text)
'Fill the address
cellPara1 = CType(row1.Cells(0).ChildEntities(5), WParagraph)
inlineCC = CType(cellPara1.ChildEntities.LastItem, InlineContentControl)
text = New WTextRange(document1)
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat)
text.Text = "2501 Aerial Center Parkway."
inlineCC.ParagraphItems.Add(text)
text = New WTextRange(document1)
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat)
text.Text = "Morrisville, NC 27560."
inlineCC.ParagraphItems.Add(text)
text = New WTextRange(document1)
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat)
text.Text = "USA."
inlineCC.ParagraphItems.Add(text)
'Fill the phone no
cellPara1 = CType(row1.Cells(0).ChildEntities(7), WParagraph)
inlineCC = CType(cellPara1.ChildEntities.LastItem, InlineContentControl)
text = New WTextRange(document1)

```

```

text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat)
text.Text = "+1 919.481.1974"
inlineCC.ParagraphItems.Add(text)
'Fill the email id
cellPara1 = CType(row1.Cells(0).ChildEntities(9), WParagraph)
inlineCC = CType(cellPara1.ChildEntities.LastItem, InlineContentControl)
text = New WTextRange(document1)
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat)
text.Text = "steve123@email.com"
inlineCC.ParagraphItems.Add(text)
#End Region
#Region "Educational Information"
table1 = CType(sec.Tables(2), WTable)
row1 = table1.Rows(1)
'Fill the education type
cellPara1 = CType(row1.Cells(0).ChildEntities(1), WParagraph)
dropDownCC = CType(cellPara1.ChildEntities.LastItem, InlineContentControl)
text = New WTextRange(document1)
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat)
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems(1).DisplayText
dropDownCC.ParagraphItems.Add(text)
'Fill the university
cellPara1 = CType(row1.Cells(0).ChildEntities(3), WParagraph)
inlineCC = CType(cellPara1.ChildEntities.LastItem, InlineContentControl)
text = New WTextRange(document1)
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat)
text.Text = "Michigan University"
inlineCC.ParagraphItems.Add(text)
'Fill the C# experience level
cellPara1 = CType(row1.Cells(0).ChildEntities(7), WParagraph)
dropDownCC = CType(cellPara1.ChildEntities.LastItem, InlineContentControl)
text = New WTextRange(document1)
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat)
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems(2).DisplayText
dropDownCC.ParagraphItems.Add(text)
'Fill the VB experience level
cellPara1 = CType(row1.Cells(0).ChildEntities(9), WParagraph)
dropDownCC = CType(cellPara1.ChildEntities.LastItem, InlineContentControl)
text = New WTextRange(document1)
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat)
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems(1).DisplayText
dropDownCC.ParagraphItems.Add(text)
#End Region
document1.Save("Form_Filled.docx")
document1.Close()

```

UWP

```

//Open the created form document
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
WordDocument document1 = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.Form_Template
.docx"), FormatType.Docx);

```

```

IWSection sec = document1.LastSection;
InlineContentControl inlineCC;
InlineContentControl dropDownCC;
WTable table1 = sec.Tables[1] as WTable;
WTableRow row1 = table1.Rows[1];
#region General Information
//Fill the name
WParagraph cellPara1 = row1.Cells[0].ChildEntities[1] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
WTextRange text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "Steve Jobs";
inlineCC.ParagraphItems.Add(text);
//Fill the date of birth
cellPara1 = row1.Cells[0].ChildEntities[3] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "06/01/1994";
inlineCC.ParagraphItems.Add(text);
//Fill the address
cellPara1 = row1.Cells[0].ChildEntities[5] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "2501 Aerial Center Parkway.";
inlineCC.ParagraphItems.Add(text);
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "Morrisville, NC 27560.";
inlineCC.ParagraphItems.Add(text);
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "USA.";
inlineCC.ParagraphItems.Add(text);
//Fill the phone no
cellPara1 = row1.Cells[0].ChildEntities[7] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "+1 919.481.1974";
inlineCC.ParagraphItems.Add(text);
//Fill the email id
cellPara1 = row1.Cells[0].ChildEntities[9] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "steve123@email.com";
inlineCC.ParagraphItems.Add(text);
#endregion
#region Educational Information
table1 = sec.Tables[2] as WTable;
row1 = table1.Rows[1];
//Fill the education type
cellPara1 = row1.Cells[0].ChildEntities[1] as WParagraph;
dropDownCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);

```

```

text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems[1].DisplayText;
dropDownCC.ParagraphItems.Add(text);
//Fill the university
cellPara1 = row1.Cells[0].ChildEntities[3] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text = "Michigan University";
inlineCC.ParagraphItems.Add(text);
//Fill the C# experience level
cellPara1 = row1.Cells[0].ChildEntities[7] as WParagraph;
dropDownCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems[2].DisplayText;
dropDownCC.ParagraphItems.Add(text);
//Fill the VB experience level
cellPara1 = row1.Cells[0].ChildEntities[9] as WParagraph;
dropDownCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems[1].DisplayText;
dropDownCC.ParagraphItems.Add(text);
#endregion
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document1.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Form_Filled.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
document1.Close();

```

ASP.NET CORE

```

//Open the created form document
WordDocument document1 = new WordDocument(outputStream,
FormatTypeAutomatic);
IWSection sec = document1.LastSection;
InlineContentControl inlineCC;
InlineContentControl dropDownCC;
WTable table1 = sec.Tables[1] as WTable;
WTableRow row1 = table1.Rows[1];
#region General Information
//Fill the name
WParagraph cellPara1 = row1.Cells[0].ChildEntities[1] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
WTextRange text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "Steve Jobs";
inlineCC.ParagraphItems.Add(text);

```

```

//Fill the date of birth
cellPara1 = row1.Cells[0].ChildEntities[3] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "06/01/1994";
inlineCC.ParagraphItems.Add(text);
//Fill the address
cellPara1 = row1.Cells[0].ChildEntities[5] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "2501 Aerial Center Parkway.";
inlineCC.ParagraphItems.Add(text);
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "Morrisville, NC 27560.";
inlineCC.ParagraphItems.Add(text);
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "USA.";
inlineCC.ParagraphItems.Add(text);
//Fill the phone no
cellPara1 = row1.Cells[0].ChildEntities[7] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "+1 919.481.1974";
inlineCC.ParagraphItems.Add(text);
//Fill the email id
cellPara1 = row1.Cells[0].ChildEntities[9] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "steve123@email.com";
inlineCC.ParagraphItems.Add(text);
#endregion
#region Educational Information
table1 = sec.Tables[2] as WTable;
row1 = table1.Rows[1];
//Fill the education type
cellPara1 = row1.Cells[0].ChildEntities[1] as WParagraph;
dropDownCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems[1].DisplayText;
dropDownCC.ParagraphItems.Add(text);
//Fill the university
cellPara1 = row1.Cells[0].ChildEntities[3] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text = "Michigan University";
inlineCC.ParagraphItems.Add(text);
//Fill the C# experience level
cellPara1 = row1.Cells[0].ChildEntities[7] as WParagraph;

```



```

dropDownCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems[2].DisplayText;
dropDownCC.ParagraphItems.Add(text);
//Fill the VB experience level
cellPara1 = row1.Cells[0].ChildEntities[9] as WParagraph;
dropDownCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems[1].DisplayText;
dropDownCC.ParagraphItems.Add(text);
#endregion
//Creates memory stream
MemoryStream saveStream = new MemoryStream();
//Saves and closes the Word document instance
document1.Save(saveStream, FormatType.Docx);
document1.Close();

```

XAMARIN

```

//Open the created form document
Stream inputStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Form_Template.docx");
WordDocument document1 = new WordDocument(inputStream,
FormatType.Automatic);
IWSection sec = document1.LastSection;
InlineContentControl inlineCC;
InlineContentControl dropDownCC;
WTable table1 = sec.Tables[1] as WTable;
WTableRow row1 = table1.Rows[1];
#region General Information
//Fill the name
WParagraph cellPara1 = row1.Cells[0].ChildEntities[1] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
WTextRange text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "Steve Jobs";
inlineCC.ParagraphItems.Add(text);
//Fill the date of birth
cellPara1 = row1.Cells[0].ChildEntities[3] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "06/01/1994";
inlineCC.ParagraphItems.Add(text);
//Fill the address
cellPara1 = row1.Cells[0].ChildEntities[5] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "2501 Aerial Center Parkway.";
inlineCC.ParagraphItems.Add(text);

```

```

text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "Morrisville, NC 27560.";
inlineCC.ParagraphItems.Add(text);
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "USA.";
inlineCC.ParagraphItems.Add(text);
//Fill the phone no
cellPara1 = row1.Cells[0].ChildEntities[7] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "+1 919.481.1974";
inlineCC.ParagraphItems.Add(text);
//Fill the email id
cellPara1 = row1.Cells[0].ChildEntities[9] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(inlineCC.BreakCharacterFormat);
text.Text = "steve123@email.com";
inlineCC.ParagraphItems.Add(text);
#endregion
#region Educational Information
table1 = sec.Tables[2] as WTable;
row1 = table1.Rows[1];
//Fill the education type
cellPara1 = row1.Cells[0].ChildEntities[1] as WParagraph;
dropDownCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems[1].DisplayText;
dropDownCC.ParagraphItems.Add(text);
//Fill the university
cellPara1 = row1.Cells[0].ChildEntities[3] as WParagraph;
inlineCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text = "Michigan University";
inlineCC.ParagraphItems.Add(text);
//Fill the C# experience level
cellPara1 = row1.Cells[0].ChildEntities[7] as WParagraph;
dropDownCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems[2].DisplayText;
dropDownCC.ParagraphItems.Add(text);
//Fill the VB experience level
cellPara1 = row1.Cells[0].ChildEntities[9] as WParagraph;
dropDownCC = cellPara1.ChildEntities.LastItem as InlineContentControl;
text = new WTextRange(document1);
text.ApplyCharacterFormat(dropDownCC.BreakCharacterFormat);
text.Text =
dropDownCC.ContentControlProperties.ContentControlListItems[1].DisplayText;
dropDownCC.ParagraphItems.Add(text);

```

```

#endregion
//Creates memory stream
MemoryStream saveStream = new MemoryStream();
//Saves and closes the Word document instance
document1.Save(saveStream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", outputStream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document1.Close();

```

Data Binding with Content Controls (XML Mapping)

Word allows you to store XML data, named *custom XML parts*, in a Word document. You can control the display of this data by binding content controls to elements in a custom XML part. When you open the Word document, the content controls display the values of the XML elements. Any changes that you make to the text in the content controls are saved in the custom XML part (two-way data binding). Refer to the following code sample to map custom XML parts to content control.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new XmlPart to the document
CustomXMLPart xmlPart = new CustomXMLPart(document);
//Loads the xml code
xmlPart.LoadXML(@"<books><book><author>Matt Hank</author><title>New
Migration Paths of the Red Breasted Robin</title><genre>New non-
fiction</genre><price>29.95</price><pub_date>12/1/2007</pub_date>
<abstract>New You see them in the spring outside your
windows.</abstract></book></books>");
//Adds the text
paragraph.AppendText("Book author name : ");
//Adds new content control to the paragraph
InlineContentControl control =
paragraph.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
//Creates the XML mapping on a content control for specified XPath
control.ContentControlProperties.XmlMapping.SetMapping("/books/book/author",
"", xmlPart);
//Selects the single node
CustomXMLNode node = xmlPart.SelectSingleNode("/books/book/title");
//Adds another paragraph
paragraph = section.AddParagraph();
//Adds text
paragraph.AppendText("Book title: ");
//Appends content control to second paragraph
control = paragraph.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;

```

```
//Creates the XML data mapping on a content control for specified node
control.ContentControlProperties.XmlMapping.SetMappingByNode (node);
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds new section to the document
Dim section As IWSection = document.AddSection
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph
'Adds new XmlPart to the document
Dim xmlPart As CustomXMLPart = New CustomXMLPart(document)
'Loads the xml code
xmlPart.LoadXML("<books><book><author>Matt Hank</author><title>New Migration Paths of the Red Breasted Robin</title><genre>New non-fiction</genre><price>29.95</price><pub_date>12/1/2007</pub_date><abstract>New You see them in the spring outside your windows.</abstract></book></books>")
'Adds text
paragraph.AppendText("Book author name : ")
'Adds new content control to the paragraph
Dim control As InlineContentControl =
CType(paragraph.AppendInlineContentControl(ContentControlType.Text),
InlineContentControl)
'Creates the XML mapping on a content control for specified XPath
control.ContentControlProperties.XmlMapping.SetMapping("/books/book/author",
"", xmlPart)
'Selects the single node
CustomXMLNode node = xmlPart.SelectSingleNode("/books/book/title");
'Adds another paragraph
paragraph = section.AddParagraph
'Adds text
paragraph.AppendText("Book title: ");
'Appends content control to second paragraph
control =
CType(paragraph.AppendInlineContentControl(ContentControlType.Text),
InlineContentControl)
'Creates the XML data mapping on a content control for specified node
control.ContentControlProperties.XmlMapping.SetMappingByNode (node)
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new XmlPart to the document.
```

```

CustomXMLPart xmlPart = new CustomXMLPart(document);
//Loads the xml code
xmlPart.LoadXML(@"<books><book><author>Matt Hank</author><title>New
Migration Paths of the Red Breasted Robin</title><genre>New non-
fiction</genre><price>29.95</price><pub_date>12/1/2007</pub_date>
<abstract>New You see them in the spring outside your
windows.</abstract></book></books>");
//Adds the text
paragraph.AppendText("Book author name : ");
//Adds new content control to the paragraph
InlineContentControl control =
paragraph.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
//Creates the XML mapping on a content control for specified XPath
control.ContentControlProperties.XmlMapping.SetMapping("/books/book/author",
"", xmlPart);
//Selects the single node
CustomXMLNode node = xmlPart.SelectSingleNode("/books/book/title");
//Adds another paragraph
paragraph = section.AddParagraph();
//Adds text
paragraph.AppendText("Book title: ");
//Appends content control to second paragraph
control = paragraph.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
//Creates the XML data mapping on a content control for specified node
control.ContentControlProperties.XmlMapping.SetMappingByNode(node);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
document.Close();

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new XmlPart to the document
CustomXMLPart xmlPart = new CustomXMLPart(document);
//Loads the xml code
xmlPart.LoadXML(@"<books><book><author>Matt Hank</author><title>New
Migration Paths of the Red Breasted Robin</title><genre>New non-
fiction</genre><price>29.95</price><pub_date>12/1/2007</pub_date>
<abstract>New You see them in the spring outside your
windows.</abstract></book></books>");
//Adds text
paragraph.AppendText("Book author name : ");
//Adds new content control to the paragraph

```

```

InlineContentControl control =
paragraph.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
//Creates the XML mapping on a content control for specified XPath
control.ContentControlProperties.XmlMapping.SetMapping("/books/book/author",
"", xmlPart);
//Selects the single node
CustomXMLNode node = xmlPart.SelectSingleNode("/books/book/title");
//Adds another paragraph
paragraph = section.AddParagraph();
//Adds text
paragraph.AppendText("Book title: ");
//Appends content control to second paragraph
control = paragraph.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
//Creates the XML data mapping on a content control for specified node
control.ContentControlProperties.XmlMapping.SetMappingByNode(node);
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
document.Close();

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new XmlPart to the document
CustomXMLPart xmlPart = new CustomXMLPart(document);
//Loads the xml code
xmlPart.LoadXML(@"<books><book><author>Matt Hank</author><title>New
Migration Paths of the Red Breasted Robin</title><genre>New non-
fiction</genre><price>29.95</price><pub_date>12/1/2007</pub_date>
<abstract>New You see them in the spring outside your
windows.</abstract></book></books>");
//Adds new content control to the paragraph
InlineContentControl control =
paragraph.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
//Creates the XML mapping on a content control for specified XPath
control.ContentControlProperties.XmlMapping.SetMapping("/books/book/author",
"", xmlPart);
//Selects the single node
CustomXMLNode node = xmlPart.SelectSingleNode("/books/book/title");
//Adds another paragraph
paragraph = section.AddParagraph();
//Adds text
paragraph.AppendText("Book title: ");
//Appends content control to second paragraph
control = paragraph.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
//Creates the XML data mapping on a content control for specified node

```

```
control.ContentControlProperties.XmlMapping.SetMappingByNode (node);
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", outputStream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document.Close();
```

Types of Content Controls

The following types of content controls can be created by using the Essential DocIO.

- Rich Text
- Plain Text
- Check Box
- Date picker
- Drop-Down List and Combo Box
- Picture

Rich Text

A rich text content control contains text or other items, such as tables, pictures, or other content controls. The following code illustrates how to add new rich text content control.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph.
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends rich text content control to the paragraph
InlineContentControl richTextControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Rich text content control.";
//Adds new text to the rich text content control
richTextControl.ParagraphItems.Add(textRange);
WPicture picture = new WPicture(document);
picture.LoadImage(Image.FromFile("Image.png"));
picture.Height = 100;
picture.Width = 100;
//Adds new picture to the rich text content control
richTextControl.ParagraphItems.Add(picture);
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
```

```
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Adds new paragraph to the section
Dim paragraph As WParagraph = document.LastParagraph
'Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ")
'Appends rich text content control to the paragraph
Dim richTextControl As InlineContentControl =
CType(paragraph.AppendInlineContentControl(ContentControlType.RichText),
InlineContentControl)
Dim textRange As WTextRange = New WTextRange(document)
textRange.Text = "Rich text content control."
'Adds new text to the rich text content control
richTextControl.ParagraphItems.Add(textRange)
Dim picture As WPicture = New WPicture(document)
picture.LoadImage(Image.FromFile("Image.png"))
picture.Height = 100
picture.Width = 100
'Adds new picture to the rich text content control
richTextControl.ParagraphItems.Add(picture)
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends rich text content control to the paragraph
InlineContentControl richTextControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Rich text content control.";
//Adds new text to the rich text content control
richTextControl.ParagraphItems.Add(textRange);
WPicture picture = new WPicture(document);
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream pictureStream =
assembly.GetManifestResourceStream("Sample.Assets.Image.png");
picture.LoadImage(pictureStream);
picture.Height = 100;
picture.Width = 100;
//Adds new picture to the rich text content control
```



```
richTextControl.ParagraphItems.Add(picture);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends rich text content control to the paragraph
InlineContentControl richTextControl =
paragraph.AppendInlineContentControl(ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Rich text content control.";
//Adds new text to the rich text content control
richTextControl.ParagraphItems.Add(textRange);
WPicture picture = new WPicture(document);
Stream imageStream = new FileStream(@"D:\Image.png", FileMode.Open,
FileAccess.Read);
//Adds image from stream
picture.LoadImage(imageStream);
picture.Height = 100;
picture.Width = 100;
//Adds new picture to the rich text content control
richTextControl.ParagraphItems.Add(picture);
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
document.Close();
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends rich text content control to the paragraph
```

```

InlineContentControl richTextControl =
paragraph.AppendInlineContentControl (ContentControlType.RichText) as
InlineContentControl;
WTextRange textRange = new WTextRange (document);
textRange.Text = "Rich text content control.";
//Adds new text to the rich text content control
richTextControl.ParagraphItems.Add (textRange);
WPicture picture = new WPicture (document);
Stream imageStream =
typeof (App).GetTypeInfo().Assembly.GetManifestResourceStream ("Sample.Assets.
Image.png");
//Adds image from stream
picture.LoadImage (imageStream);
picture.Height = 100;
picture.Width = 100;
//Adds new picture to the rich text content control
richTextControl.ParagraphItems.Add (picture);
//Creates memory stream
MemoryStream outputStream = new MemoryStream ();
//Saves and closes the Word document instance
document.Save (outputStream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView ("Result.docx",
"application/msword", outputStream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document.Close ();

```

Plain Text

A plain text content control contains text and cannot contain other items, such as tables, pictures, or other content controls. Refer to the following code to add plain text content control.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument ();
//Adds one section and one paragraph to the document
document.EnsureMinimal ();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText ("A new text is added to the paragraph. ");
//Appends plain text content control to the paragraph
InlineContentControl plainTextControl =
paragraph.AppendInlineContentControl (ContentControlType.Text) as
InlineContentControl;
WTextRange textRange = new WTextRange (document);
textRange.Text = "Plain text content control.";
//Adds new text to the plain text content control
plainTextControl.ParagraphItems.Add (textRange);
//Saves and closes the Word document instance
document.Save ("Sample.docx", FormatType.Docx);
document.Close ();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Adds new paragraph to the section
Dim paragraph As WParagraph = document.LastParagraph
'Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ")
'Appends plain text content control to the paragraph
Dim plainTextControl As InlineContentControl =
CType(paragraph.AppendInlineContentControl(ContentControlType.Text),
InlineContentControl)
Dim textRange As WTextRange = New WTextRange(document)
textRange.Text = "Plain text content control."
'Adds new text to the plain text content control
plainTextControl.ParagraphItems.Add(textRange)
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends plain text content control to the paragraph
InlineContentControl plainTextControl =
paragraph.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Plain text content control.";
//Adds new text to the plain text content control
plainTextControl.ParagraphItems.Add(textRange);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
document.Close(); //Please refer the below link to save Word document in UWP
platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();

```

```
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends plain text content control to the paragraph
InlineContentControl plainTextControl =
paragraph.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Plain text content control.";
//Adds new text to the plain text content control
plainTextControl.ParagraphItems.Add(textRange);
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
document.Close();
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends plain text content control to the paragraph
InlineContentControl plainTextControl =
paragraph.AppendInlineContentControl(ContentControlType.Text) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
textRange.Text = "Plain text content control.";
//Adds new text to the plain text content control
plainTextControl.ParagraphItems.Add(textRange);
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", outputStream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document.Close();
```

Check Box

A check box content control provides a UI that represents a binary state: checked or unchecked. Default state for check box is unchecked. Refer to the following code to add check box content control.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph.
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends checkbox content control to the paragraph
InlineContentControl checkBox =
paragraph.AppendInlineContentControl(ContentControlType.CheckBox) as
InlineContentControl;
checkBox.ContentControlProperties.IsChecked = true;
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Adds new paragraph to the section
Dim paragraph As WParagraph = document.LastParagraph
'Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ")
'Appends checkbox content control to the paragraph
Dim checkBox As InlineContentControl =
CType(paragraph.AppendInlineContentControl(ContentControlType.CheckBox),
InlineContentControl)
checkBox.ContentControlProperties.IsChecked = True
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends checkbox content control to the paragraph
InlineContentControl checkBox =
paragraph.AppendInlineContentControl(ContentControlType.CheckBox) as
InlineContentControl;
checkBox.ContentControlProperties.IsChecked = true;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");

```

```
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends picture content control to the paragraph
InlineContentControl checkBox =
paragraph.AppendInlineContentControl(ContentControlType.CheckBox) as
InlineContentControl;
checkBox.ContentControlProperties.IsChecked = true;
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
document.Close();
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends picture content control to the paragraph
InlineContentControl checkBox =
paragraph.AppendInlineContentControl(ContentControlType.CheckBox) as
InlineContentControl;
checkBox.ContentControlProperties.IsChecked = true;
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", outputStream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-xamarin#helper-files-for-xamarin
document.Close();
```

Date picker

A date picker content control provides a calendar UI for selecting a date. The calendar appears when you click the drop-down arrow in the content control. You can use regional calendars and different date formats. Refer to the following code to add date picker content control.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("Select Date: ");
//Appends date picker content control to the paragraph
InlineContentControl datePicker =
paragraph.AppendInlineContentControl(ContentControlType.Date) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
//Sets today's date to display
textRange.Text = DateTime.Now.ToShortDateString();
datePicker.ParagraphItems.Add(textRange);
//Sets calendar type for the date picker content control
datePicker.ContentControlProperties.DateCalendarType =
CalendarType.Gregorian;
//Sets the format for date to display
datePicker.ContentControlProperties.DateDisplayFormat = "M/d/yyyy";
//Sets the language format for the date
datePicker.ContentControlProperties.DateDisplayLocale = LocaleIDs.en_US;
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Adds new paragraph to the section
Dim paragraph As WParagraph = document.LastParagraph
'Adds text to the paragraph
paragraph.AppendText("Select Date: ")
'Appends date picker content control to the paragraph
Dim datePicker As InlineContentControl =
CType(paragraph.AppendInlineContentControl(ContentControlType.Date),
InlineContentControl)
Dim textRange As WTextRange = New WTextRange(document)
'Sets today's date to display
textRange.Text = DateTime.Now.ToShortDateString()
datePicker.ParagraphItems.Add(textRange)
'Sets calendar type for the date picker content control
datePicker.ContentControlProperties.DateCalendarType =
CalendarType.Gregorian
'Sets the format for date to display
```

```

datePicker.ContentControlProperties.DateDisplayFormat = "M/d/yyyy"
'Sets the language format for the date
datePicker.ContentControlProperties.DateDisplayLocale = LocaleIDs.en_US
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("Select Date: ");
//Appends date picker content control to the paragraph
InlineContentControl datePicker =
paragraph.AppendInlineContentControl(ContentControlType.Date) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
//Sets today's date to display
textRange.Text = DateTime.Now.ToShortDateString();
datePicker.ParagraphItems.Add(textRange);
//Sets calendar type for the date picker content control
datePicker.ContentControlProperties.DateCalendarType =
CalendarType.Gregorian;
//Sets the format for date to display
datePicker.ContentControlProperties.DateDisplayFormat = "M/d/yyyy";
//Sets the language format for the date
datePicker.ContentControlProperties.DateDisplayLocale = LocaleIDs.en_US;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("Select Date: ");
//Appends date picker content control to the paragraph
InlineContentControl datePicker =
paragraph.AppendInlineContentControl(ContentControlType.Date) as
InlineContentControl;

```



```

WTextRange textRange = new WTextRange(document);
//Sets today's date to display
textRange.Text = DateTime.Now.ToString();
datePicker.ParagraphItems.Add(textRange);
//Sets calendar type for the date picker content control
datePicker.ContentControlProperties.DateCalendarType =
CalendarType.Gregorian;
//Sets the format for date to display
datePicker.ContentControlProperties.DateDisplayFormat = "M/d/yyyy";
//Sets the language format for the date
datePicker.ContentControlProperties.DateDisplayLocale = LocaleIDs.en_US;
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
document.Close();

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("Select Date: ");
//Appends date picker content control to the paragraph
InlineContentControl datePicker =
paragraph.AppendInlineContentControl(ContentControlType.Date) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
//Sets today's date to display
textRange.Text = DateTime.Now.ToString();
datePicker.ParagraphItems.Add(textRange);
//Sets calendar type for the date picker content control
datePicker.ContentControlProperties.DateCalendarType =
CalendarType.Gregorian;
//Sets the format for date to display
datePicker.ContentControlProperties.DateDisplayFormat = "M/d/yyyy";
//Sets the language format for the date
datePicker.ContentControlProperties.DateDisplayLocale = LocaleIDs.en_US;
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", outputStream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document.Close();

```

Drop-Down List and Combo Box

A drop-down list content control and combo box content control displays a list of items you can select. Unlike a drop-down list, the combo box allows to add your own items. Refer to the following code to add drop-down list and combo box content controls.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Appends dropdown list content control to the paragraph
InlineContentControl dropdown =
paragraph.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
//Sets default option to display
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
ContentControlListItem item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "ASP.NET MVC";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Windows Forms";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "WPF";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
//Adds new paragraph to the section
paragraph = section.AddParagraph() as WParagraph;
//Appends combo box content control to the paragraph
InlineContentControl comboBox =
paragraph.AppendInlineContentControl(ContentControlType.ComboBox) as
InlineContentControl;
textRange = new WTextRange(document);
//Sets default option to display
textRange.Text = "Choose an item from combo box";
comboBox.ParagraphItems.Add(textRange);
//Creates an item for combo box
item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Word to HTML";
//Sets the value to the list item
item.Value = "1";
comboBox.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Word to Image";
item.Value = "2";
```

```

comboBox.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Word to PDF";
item.Value = "3";
comboBox.ContentControlProperties.ContentControlListItems.Add(item);
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds new section to the document
Dim section As IWSection = document.AddSection
'Adds new paragraph to the section
Dim paragraph As WParagraph = CType(section.AddParagraph, WParagraph)
'Appends dropdown list content control to the paragraph
Dim dropdown As InlineContentControl =
CType(paragraph.AppendInlineContentControl(ContentControlType.DropDownList),
InlineContentControl)
Dim textRange As WTextRange = New WTextRange(document)
'Sets default option to display
textRange.Text = "Choose an item from drop down list"
dropdown.ParagraphItems.Add(textRange)
'Creates an item for dropdown list
Dim item As ContentControlListItem = New ContentControlListItem
'Sets the text to be displayed as list item
item.DisplayText = "ASP.NET MVC"
'Sets the value to the list item
item.Value = "1"
'Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item)
item = New ContentControlListItem
item.DisplayText = "Windows Forms"
item.Value = "2"
dropdown.ContentControlProperties.ContentControlListItems.Add(item)
item = New ContentControlListItem
item.DisplayText = "WPF"
item.Value = "3"
dropdown.ContentControlProperties.ContentControlListItems.Add(item)
'Adds new paragraph to the section
paragraph = CType(section.AddParagraph, WParagraph)
'Appends combo box content control to the paragraph
Dim comboBox As InlineContentControl =
CType(paragraph.AppendInlineContentControl(ContentControlType.ComboBox),
InlineContentControl)
textRange = New WTextRange(document)
'Sets default option to display
textRange.Text = "Choose an item from combo box"
comboBox.ParagraphItems.Add(textRange)
'Creates an item for combo box
item = New ContentControlListItem
'Sets the text to be displayed as list item
item.DisplayText = "Word to HTML"
'Sets the value to the list item

```

```

item.Value = "1"
comboBox.ContentControlProperties.ContentControlListItems.Add(item)
item = New ContentControlListItem
item.DisplayText = "Word to Image"
item.Value = "2"
comboBox.ContentControlProperties.ContentControlListItems.Add(item)
item = New ContentControlListItem
item.DisplayText = "Word to PDF"
item.Value = "3"
comboBox.ContentControlProperties.ContentControlListItems.Add(item)
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Appends dropdown list content control to the paragraph
InlineContentControl dropdown =
paragraph.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
//Sets default option to display
textRange.Text = "Choose an item from drop down list";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
ContentControlListItem item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "ASP.NET MVC";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Windows Forms";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "WPF";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
//Adds new paragraph to the section
paragraph = section.AddParagraph() as WParagraph;
//Appends combo box content control to the paragraph
InlineContentControl comboBox =
paragraph.AppendInlineContentControl(ContentControlType.ComboBox) as
InlineContentControl;
textRange = new WTextRange(document);
//Sets default option to display
textRange.Text = "Choose an item from combo box";
comboBox.ParagraphItems.Add(textRange);

```

```
//Creates an item for combo box
item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Word to HTML";
//Sets the value to the list item
item.Value = "1";
comboBox.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Word to Image";
item.Value = "2";
comboBox.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Word to PDF";
item.Value = "3";
comboBox.ContentControlProperties.ContentControlListItems.Add(item);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Adds text to the paragraph
paragraph.AppendText("Choose your platform: ");
//Appends dropdown list content control to the paragraph
InlineContentControl dropdown =
paragraph.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
//Sets default option to display
textRange.Text = "Choose an item";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
ContentControlListItem item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "ASP.NET MVC";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Windows Forms";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
```

```

item.DisplayText = "WPF";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
//Adds new paragraph to the section
paragraph = section.AddParagraph() as WParagraph;
//Adds text to the paragraph
paragraph.AppendText("Choose the conversion: ");
//Appends combo box content control to the paragraph
InlineContentControl comboBox =
paragraph.AppendInlineContentControl(ContentControlType.ComboBox) as
InlineContentControl;
textRange = new WTextRange(document);
//Sets default option to display
textRange.Text = "Choose an item";
comboBox.ParagraphItems.Add(textRange);
//Creates an item for combo box
item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Word to HTML";
//Sets the value to the list item
item.Value = "1";
comboBox.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Word to Image";
item.Value = "2";
comboBox.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Word to PDF";
item.Value = "3";
comboBox.ContentControlProperties.ContentControlListItems.Add(item);
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
document.Close();

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
WParagraph paragraph = section.AddParagraph() as WParagraph;
//Adds text to the paragraph
paragraph.AppendText("Choose your platform: ");
//Appends dropdown list content control to the paragraph
InlineContentControl dropdown =
paragraph.AppendInlineContentControl(ContentControlType.DropDownList) as
InlineContentControl;
WTextRange textRange = new WTextRange(document);
//Sets default option to display
textRange.Text = "Choose an item";
dropdown.ParagraphItems.Add(textRange);
//Creates an item for dropdown list
ContentControlListItem item = new ContentControlListItem();

```

```

//Sets the text to be displayed as list item
item.DisplayText = "ASP.NET MVC";
//Sets the value to the list item
item.Value = "1";
//Adds item to the dropdown list
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Windows Forms";
item.Value = "2";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "WPF";
item.Value = "3";
dropdown.ContentControlProperties.ContentControlListItems.Add(item);
//Adds new paragraph to the section
paragraph = section.AddParagraph() as WParagraph;
//Adds text to the paragraph
paragraph.AppendText("Choose the conversion: ");
//Appends combo box content control to the paragraph
InlineContentControl comboBox =
paragraph.AppendInlineContentControl(ContentControlType.ComboBox) as
InlineContentControl;
textRange = new WTextRange(document);
//Sets default option to display
textRange.Text = "Choose an item";
comboBox.ParagraphItems.Add(textRange);
//Creates an item for combo box
item = new ContentControlListItem();
//Sets the text to be displayed as list item
item.DisplayText = "Word to HTML";
//Sets the value to the list item
item.Value = "1";
comboBox.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Word to Image";
item.Value = "2";
comboBox.ContentControlProperties.ContentControlListItems.Add(item);
item = new ContentControlListItem();
item.DisplayText = "Word to PDF";
item.Value = "3";
comboBox.ContentControlProperties.ContentControlListItems.Add(item);
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", outputStream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document.Close();

```

Picture

A picture content control displays an image. Refer to the following code to add new picture content control.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph.
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends picture content control to the paragraph
InlineContentControl pictureContentControl =
paragraph.AppendInlineContentControl(ContentControlType.Picture) as
InlineContentControl;
//Creates a new image instance and load image
WPicture picture = new WPicture(document);
picture.LoadImage(Image.FromFile("Image.png"));
//Adds picture to the picture content control
pictureContentControl.ParagraphItems.Add(picture);
//Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Adds new paragraph to the section
Dim paragraph As WParagraph = document.LastParagraph
'Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ")
'Appends picture content control to the paragraph
Dim pictureContentControl As InlineContentControl =
CType(paragraph.AppendInlineContentControl(ContentControlType.Picture),
InlineContentControl)
'Creates a new image instance and load image
Dim picture As WPicture = New WPicture(document)
picture.LoadImage(Image.FromFile("Image.png"))
'Adds picture to the picture content control
pictureContentControl.ParagraphItems.Add(picture)
'Saves and closes the Word document instance
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
```



```

//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends picture content control to the paragraph
InlineContentControl pictureContentControl =
paragraph.AppendInlineContentControl(ContentControlType.Picture) as
InlineContentControl;
//Creates a new image instance and load image
WPicture picture = new WPicture(document);
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream pictureStream =
assembly.GetManifestResourceStream("Sample.Assets.Image.png");
picture.LoadImage(pictureStream);
//Adds picture to the picture content control
pictureContentControl.ParagraphItems.Add(picture);
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends picture content control to the paragraph
InlineContentControl pictureContentControl =
paragraph.AppendInlineContentControl(ContentControlType.Picture) as
InlineContentControl;
//Creates a new image instance and load image
WPicture picture = new WPicture(document);
Stream imageStream = new FileStream(@"D:\Image.png", FileMode.Open,
FileAccess.Read);
//Adds image from stream
picture.LoadImage(imageStream);
//Adds picture to the picture content control
pictureContentControl.ParagraphItems.Add(picture);
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
document.Close();

```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Gets the last paragraph
WParagraph paragraph = document.LastParagraph;
//Adds text to the paragraph
paragraph.AppendText("A new text is added to the paragraph. ");
//Appends picture content control to the paragraph
InlineContentControl pictureContentControl =
paragraph.AppendInlineContentControl(ContentControlType.Picture) as
InlineContentControl;
//Creates a new image instance and load image
WPicture picture = new WPicture(document);
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Image.png");
//Adds image from stream
picture.LoadImage(imageStream);
//Adds picture to the picture content control
pictureContentControl.ParagraphItems.Add(picture);
//Creates memory stream
MemoryStream outputStream = new MemoryStream();
//Saves and closes the Word document instance
document.Save(outputStream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Result.docx",
"application/msword", outputStream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
document.Close();
```

Note: In the above-mentioned code samples, for Xamarin platforms the document is saved as stream only. To save the stream to file kindly refer code sample [here](#).

Working with Mathematical Equation

Equations in Word document are combination of mathematical symbols or text. For example, you can create a Fourier series equation in Word document.

![Mathematical equation in Microsoft Word document](WorkingwithMathematicalEquation_images/Mathematical Equation.png)

Note: You can use mathematical equation only in documents that are saved in the Open XML Format and cannot be used in the Word 97-2003 document (.doc) format.

Types of equation

The following different structures of equation can be created by using the Essential DocIO.



- Accent
- Bar
- Box
- Border box
- Delimiter
- Equation array
- Fraction
- Function
- Group character
- Limit
- Matrix
- N-Array
- Radical
- Phantom
- SubSuperscript
- Left SubSuperscript
- Right SubSuperscript

Accent

You can add an accent mark to the equation. The following code example shows how to add an accent mark to the equation.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds an accent equation
IOfficeMathAccent mathAccent =
officeMath.Functions.Add(MathFunctionType.Accent) as IOfficeMathAccent;
//Sets the accent character
mathAccent.AccentCharacter = "~";
//Adds the run element for accent
IOfficeMathRunElement officeMathRunElement =
mathAccent.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
WTextRange textRange = officeMathRunElement.Item as WTextRange;
//Sets text for accent equation
textRange.Text = "a";
//Applies character formatting for text range
textRange.CharacterFormat.Bold = true;
textRange.CharacterFormat.Italic = true;
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOfficeMath = math.MathParagraph.Maths.Add
'Adds an accent equation
Dim mathAccent As IOfficeMathAccent =
CType(officeMath.Functions.Add(MathFunctionType.Accent), IOfficeMathAccent)
'Sets the accent character
mathAccent.AccentCharacter = ""
Dim officeMathRunElement As IOfficeMathRunElement =
CType(mathAccent.Equation.Functions.Add(MathFunctionType.RunElement),
IOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
Dim textRange As WTextRange = CType(officeMathRunElement.Item, WTextRange)
'Sets text for accent equation
textRange.Text = "a"
'Applies character formatting for text range
textRange.CharacterFormat.Bold = True
textRange.CharacterFormat.Italic = True
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds an accent equation
IOfficeMathAccent mathAccent =
officeMath.Functions.Add(MathFunctionType.Accent) as IOfficeMathAccent;
//Sets the accent character
mathAccent.AccentCharacter = "°";
//Adds the run element for accent
IOfficeMathRunElement officeMathRunElement =
mathAccent.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
WTextRange textRange = officeMathRunElement.Item as WTextRange;
//Sets text for accent equation
textRange.Text = "a";
//Applies character formatting for text range
textRange.CharacterFormat.Bold = true;
textRange.CharacterFormat.Italic = true;

```

```
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx ");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds an accent equation
IOfficeMathAccent mathAccent =
officeMath.Functions.Add(MathFunctionType.Accent) as IOfficeMathAccent;
//Sets the accent character
mathAccent.AccentCharacter = "~";
//Adds the run element for accent
IOfficeMathRunElement officeMathRunElement =
mathAccent.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
WTextRange textRange = officeMathRunElement.Item as WTextRange;
//Sets text for accent equation
textRange.Text = "a";
//Applies character formatting for text range
textRange.CharacterFormat.Bold = true;
textRange.CharacterFormat.Italic = true;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds an accent equation
```

```

IOfficeMathAccent mathAccent =
officeMath.Functions.Add(MathFunctionType.Accent) as IOfficeMathAccent;
//Sets the accent character
mathAccent.AccentCharacter = "˘";
//Adds the run element for accent
IOfficeMathRunElement officeMathRunElement =
mathAccent.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
WTextRange textRange = officeMathRunElement.Item as WTextRange;
//Sets text for accent equation
textRange.Text = "a";
//Applies character formatting for text range
textRange.CharacterFormat.Bold = true;
textRange.CharacterFormat.Italic = true;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN

```

Bar

You can add a bar (which adds horizontal line on top or bottom) to the equation. The following code example shows how to add a bar to the equation.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Add a section and a paragraph in the empty document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a bar equation
IOfficeMathBar mathBar = officeMath.Functions.Add(0, MathFunctionType.Bar)
as IOfficeMathBar;
//Sets the position of bar
mathBar.BarTop = true;
//Adds the run element for bar
IOfficeMathRunElement officeMathRunElement =
mathBar.Equation.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for bar equation
(officeMathRunElement.Item as WTextRange).Text = "a";
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);

```

```
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOOfficeMath = math.MathParagraph.Maths.Add
Dim mathBar As IOOfficeMathBar = CType(officeMath.Functions.Add(0,
MathFunctionType.Bar), IOOfficeMathBar)
'Sets the position of bar
mathBar.BarTop = True
Dim officeMathRunElement As IOOfficeMathRunElement =
CType(mathBar.Equation.Functions.Add(0, MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for bar equation
CType(officeMathRunElement.Item, WTextRange).Text = "a"
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds an bar equation
IOOfficeMathBar mathBar = officeMath.Functions.Add(0, MathFunctionType.Bar)
as IOOfficeMathBar;
//Sets the position of bar
mathBar.BarTop = true;
//Adds the run element for bar
IOOfficeMathRunElement officeMathRunElement =
mathBar.Equation.Functions.Add(0, MathFunctionType.RunElement) as
IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for bar equation
(officeMathRunElement.Item as WTextRange).Text = "a";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
```

```
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds an bar function
IOfficeMathBar mathBar = officeMath.Functions.Add(0, MathFunctionType.Bar)
as IOfficeMathBar;
//Sets the bar top
mathBar.BarTop = true;
//Adds the run element for bar
IOfficeMathRunElement officeMathRunElement =
mathBar.Equation.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for bar equation
(officeMathRunElement.Item as WTextRange).Text = "a";
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds an bar function
IOfficeMathBar mathBar = officeMath.Functions.Add(0, MathFunctionType.Bar)
as IOfficeMathBar;
//Sets the bar top
mathBar.BarTop = true;
//Adds the run element for bar
IOfficeMathRunElement officeMathRunElement =
mathBar.Equation.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for bar equation
(officeMathRunElement.Item as WTextRange).Text = "a";
//Saves the Word document to MemoryStream
```



```

MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN
document.Close();

```

Box

You can add a box to the equation. The following code example shows how to add a box to the equation.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a box equation
IOfficeMathBox mathBox = officeMath.Functions.Add(0, MathFunctionType.Box)
as IOfficeMathBox;
//Adds the run element for box
IOfficeMathRunElement officeMathRunElement =
officeMath.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for math
(officeMathRunElement.Item as WTextRange).Text = "a+b";
//Enables the flag, to behave the box and its contents as a single operator
mathBox.OperatorEmulator = true;
//Enables the flag, to act box as the mathematical differential
mathBox.EnableDifferential = true;
//Adds a break in box equation
mathBox.Break = officeMath.Breaks.Add(0);
//Adds the run element for box
officeMathRunElement =
mathBox.Equation.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for box equation
(officeMathRunElement.Item as WTextRange).Text = "==" ;
//Adds the run element for box
officeMathRunElement =
mathBox.Equation.Functions.Add(1, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for box equation
(officeMathRunElement.Item as WTextRange).Text = "adx";
//Saves the Word document

```

```
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOOfficeMath = math.MathParagraph.Maths.Add
Dim mathBox As IOOfficeMathBox = CType(officeMath.Functions.Add(0,
MathFunctionType.Box), IOOfficeMathBox)
'Adds the run element for box
Dim officeMathRunElement As IOOfficeMathRunElement =
CType(officeMath.Functions.Add(0, MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for math
CType(officeMathRunElement.Item, WTextRange).Text = "a+b"
'Enables the flag, to behave the box and its contents as a single operator
mathBox.OperatorEmulator = True
'Enables the flag, to act box as the mathematical differential
mathBox.EnableDifferential = True
'Adds a break in box equation
mathBox.Break = officeMath.Breaks.Add(0)
'Adds the run element for box
officeMathRunElement = CType(mathBox.Equation.Functions.Add(0,
MathFunctionType.RunElement), IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for box equation
CType(officeMathRunElement.Item, WTextRange).Text = "=="
'Adds the run element for box
officeMathRunElement = CType(mathBox.Equation.Functions.Add(1,
MathFunctionType.RunElement), IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for box equation
CType(officeMathRunElement.Item, WTextRange).Text = "adx"
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOOfficeMath officeMath = math.MathParagraph.Maths.Add();
```

```

//Adds a box equation
IOfficeMathBox mathBox = officeMath.Functions.Add(0, MathFunctionType.Box)
as IOfficeMathBox;
//Adds the run element for box
IOfficeMathRunElement officeMathRunElement =
officeMath.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for math
(officeMathRunElement.Item as WTextRange).Text = "a+b";
//Enables the flag, to behave the box and its contents as a single operator
mathBox.OperatorEmulator = true;
//Enables the flag, to act box as the mathematical differential
mathBox.EnabledDifferential = true;
//Adds a break in box equation
mathBox.Break = officeMath.Breaks.Add(0);
//Adds the run element for box
officeMathRunElement =
mathBox.Equation.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for box equation
(officeMathRunElement.Item as WTextRange).Text = "==" ;
//Adds the run element for box
officeMathRunElement =
mathBox.Equation.Functions.Add(1, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for box equation
(officeMathRunElement.Item as WTextRange).Text = "adx";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a box equation
IOfficeMathBox mathBox = officeMath.Functions.Add(0, MathFunctionType.Box)
as IOfficeMathBox;
//Adds the run element for box
IOfficeMathRunElement officeMathRunElement =

```

```

officeMath.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for math
(officeMathRunElement.Item as WTextRange).Text = "a+b";
//Enables the flag, to behave the box and its contents as a single operator
mathBox.OperatorEmulator = true;
//Enables the flag, to act box as the mathematical differential
mathBox.EnableDifferential = true;
//Adds a break in box equation
mathBox.Break = officeMath.Breaks.Add(0);
//Adds the run element for box
officeMathRunElement =
mathBox.Equation.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for box equation
(officeMathRunElement.Item as WTextRange).Text = "==" ;
//Adds the run element for box
officeMathRunElement =
mathBox.Equation.Functions.Add(1, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for box equation
(officeMathRunElement.Item as WTextRange).Text = "adx";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a box equation
IOfficeMathBox mathBox = officeMath.Functions.Add(0, MathFunctionType.Box)
as IOfficeMathBox;
//Adds the run element for box
IOfficeMathRunElement officeMathRunElement =
officeMath.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for math
(officeMathRunElement.Item as WTextRange).Text = "a+b";
//Enables the flag, to behave the box and its contents as a single operator
mathBox.OperatorEmulator = true;

```

```

//Enables the flag, to act box as the mathematical differential
mathBox.EnableDifferential = true;
//Adds a break in box equation
mathBox.Break = officeMath.Breaks.Add(0);
//Adds the run element for box
officeMathRunElement =
mathBox.Equation.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for box equation
(officeMathRunElement.Item as WTextRange).Text = "==" ;
//Adds the run element for box
officeMathRunElement =
mathBox.Equation.Functions.Add(1, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for box equation
(officeMathRunElement.Item as WTextRange).Text = "adx";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN

```

Border box

You can add a box with the borders on four sides and strikethrough on horizontal, vertical, and diagonal directions to the equation. The following code example shows how to add a border box to the equation.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a border box equation
IOfficeMathBorderBox mathBorderBox =
officeMath.Functions.Add(0, MathFunctionType.BorderBox) as
IOfficeMathBorderBox;
//Sets the diagonal strikethrough from lower left to upper right
mathBorderBox.StrikeDiagonalUp = true;
//Sets the diagonal strikethrough from upper left to lower right
mathBorderBox.StrikeDiagonalDown = true;
//Sets the horizontal strikethrough
mathBorderBox.StrikeHorizontal = true;
//Sets the vertical strikethrough

```

```

mathBorderBox.StrikeVertical = true;
//Enables the flag, to hide the bottom border of an equation
mathBorderBox.HideBottom = true;
//Enables the flag, to hide the left border of an equation
mathBorderBox.HideLeft = true;
//Sets false to show the right border of an equation
mathBorderBox.HideRight = false;
//Sets false to show the top border of an equation
mathBorderBox.HideTop = false;
//Adds the run element for border box
IOfficeMathRunElement officeMathRunElement =
mathBorderBox.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for border box equation
(officeMathRunElement.Item as WTextRange).Text = "a+b-c";
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOfficeMath = math.MathParagraph.Maths.Add
Dim mathBorderBox As IOfficeMathBorderBox =
CType(officeMath.Functions.Add(0, MathFunctionType.BorderBox),
IOfficeMathBorderBox)
'Sets the diagonal strikethrough from lower left to upper right
mathBorderBox.StrikeDiagonalUp = True
'Sets the diagonal strikethrough from upper left to lower right
mathBorderBox.StrikeDiagonalDown = True
'Sets the horizontal strikethrough
mathBorderBox.StrikeHorizontal = True
'Sets the vertical strikethrough
mathBorderBox.StrikeVertical = True
'Enable the flag, to hide the bottom border of an equation
mathBorderBox.HideBottom = True
'Enable the flag, to hide the left border of an equation
mathBorderBox.HideLeft = True
'Enable the flag, to hide the right border of an equation
mathBorderBox.HideRight = False
'Enable the flag, to hide the top border of an equation
mathBorderBox.HideTop = False
Dim officeMathRunElement As IOfficeMathRunElement =
CType(mathBorderBox.Equation.Functions.Add(MathFunctionType.RunElement),
IOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for border box equation
CType(officeMathRunElement.Item, WTextRange).Text = "a+b-c"

```

```
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a border box equation
IOfficeMathBorderBox mathBorderBox =
officeMath.Functions.Add(0, MathFunctionType.BorderBox) as
IOfficeMathBorderBox;
//Sets the diagonal strikethrough from lower left to upper right
mathBorderBox.StrikeDiagonalUp = true;
//Sets the diagonal strikethrough from upper left to lower right
mathBorderBox.StrikeDiagonalDown = true;
//Sets the horizontal strikethrough
mathBorderBox.StrikeHorizontal = true;
//Sets the vertical strikethrough
mathBorderBox.StrikeVertical = true;
//Enable the flag, to hide the bottom border of an equation
mathBorderBox.HideBottom = true;
//Enable the flag, to hide the left border of an equation
mathBorderBox.HideLeft = true;
//Enable the flag, to hide the right border of an equation
mathBorderBox.HideRight = false;
//Enable the flag, to hide the top border of an equation
mathBorderBox.HideTop = false;
//Adds the run element for border box
IOfficeMathRunElement officeMathRunElement =
mathBorderBox.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for border box equation
(officeMathRunElement.Item as WTextRange).Text = "a+b-c";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
```

```
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a border box equation
IOfficeMathBorderBox mathBorderBox =
officeMath.Functions.Add(0, MathFunctionType.BorderBox) as
IOfficeMathBorderBox;
//Sets the diagonal strikethrough from lower left to upper right
mathBorderBox.StrikeDiagonalUp = true;
//Sets the diagonal strikethrough from upper left to lower right
mathBorderBox.StrikeDiagonalDown = true;
//Sets the horizontal strikethrough
mathBorderBox.StrikeHorizontal = true;
//Sets the vertical strikethrough
mathBorderBox.StrikeVertical = true;
//Enable the flag, to hide the bottom border of an equation
mathBorderBox.HideBottom = true;
//Enable the flag, to hide the left border of an equation
mathBorderBox.HideLeft = true;
//Enable the flag, to hide the right border of an equation
mathBorderBox.HideRight = false;
//Enable the flag, to hide the top border of an equation
mathBorderBox.HideTop = false;
//Adds the run element for border box
IOfficeMathRunElement officeMathRunElement =
mathBorderBox.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for border box equation
(officeMathRunElement.Item as WTextRange).Text = "a+b-c";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a border box equation
IOfficeMathBorderBox mathBorderBox =
officeMath.Functions.Add(0, MathFunctionType.BorderBox) as
IOfficeMathBorderBox;
```



```

//Sets the diagonal strikethrough from lower left to upper right
mathBoundingBox.StrikeDiagonalUp = true;
//Sets the diagonal strikethrough from upper left to lower right
mathBoundingBox.StrikeDiagonalDown = true;
//Sets the horizontal strikethrough
mathBoundingBox.StrikeHorizontal = true;
//Sets the vertical strikethrough
mathBoundingBox.StrikeVertical = true;
//Enable the flag, to hide the bottom border of an equation
mathBoundingBox.HideBottom = true;
//Enable the flag, to hide the left border of an equation
mathBoundingBox.HideLeft = true;
//Enable the flag, to hide the right border of an equation
mathBoundingBox.HideRight = false;
//Enable the flag, to hide the top border of an equation
mathBoundingBox.HideTop = false;
//Adds the run element for border box
IOfficeMathRunElement officeMathRunElement =
mathBoundingBox.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for border box equation
(officeMathRunElement.Item as WTextRange).Text = "a+b-c";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN

```

Delimiter

You can add a delimiter (parenthesis, square brackets and other characters) to the equation. The following code example shows how to add a delimiter to the equation.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a delimiter equation
IOfficeMathDelimiter mathDelimiter =
officeMath.Functions.Add(0, MathFunctionType.Delimiter) as
IOfficeMathDelimiter;
//Sets the begin character
mathDelimiter.BeginCharacter = "[";

```

```

//Sets the end character
mathDelimiter.EndCharacter = "]"";
//Enables the flag, to grow delimiter characters to full height of the arguments
mathDelimiter.IsGrow = true;
//Sets the appearance of delimiters
mathDelimiter.DelimiterShape = MathDelimiterShapeType.Match;
//Adds the run element for delimiter
IOfficeMathRunElement officeMathRunElement =
mathDelimiter.Equation.Add(0).Functions.Add(0, MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for delimiter equation
(officeMathRunElement.Item as WTextRange).Text = "a+b";
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOfficeMath = math.MathParagraph.Maths.Add
Dim mathDelimiter As IOfficeMathDelimiter =
CType(officeMath.Functions.Add(0, MathFunctionType.Delimiter),
IOfficeMathDelimiter)
'Sets the begin character
mathDelimiter.BeginCharacter = "["
'Sets the end character
mathDelimiter.EndCharacter = "]"
'Enables the flag, to grow delimiter characters to full height of the arguments
mathDelimiter.IsGrow = True
'Sets the appearance of delimiters
mathDelimiter.DelimiterShape = MathDelimiterShapeType.Match
Dim officeMathRunElement As IOfficeMathRunElement =
CType(mathDelimiter.Equation.Add(0).Functions.Add(0,
MathFunctionType.RunElement), IOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for delimiter equation
CType(officeMathRunElement.Item, WTextRange).Text = "a+b"
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();

```

```

//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a delimiter equation
IOfficeMathDelimiter mathDelimiter =
officeMath.Functions.Add(0, MathFunctionType.Delimiter) as
IOfficeMathDelimiter;
//Sets the begin character
mathDelimiter.BeginCharacter = "[";
//Sets the end character
mathDelimiter.EndCharacter = "]";
//Enables the flag, to grow delimiter characters to full height of the arguments
mathDelimiter.IsGrow = true;
//Sets the appearance of delimiters
mathDelimiter.DelimiterShape = MathDelimiterShapeType.Match;
//Adds the run element for delimiter
IOfficeMathRunElement officeMathRunElement =
mathDelimiter.Equation.Add(0).Functions.Add(0, MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for delimiter equation
(officeMathRunElement.Item as WTextRange).Text = "a+b";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a delimiter equation
IOfficeMathDelimiter mathDelimiter =
officeMath.Functions.Add(0, MathFunctionType.Delimiter) as
IOfficeMathDelimiter;
//Sets the begin character
mathDelimiter.BeginCharacter = "[";
//Sets the end character
mathDelimiter.EndCharacter = "]";
//Enables the flag, to grow delimiter characters to full height of the arguments

```

```

mathDelimiter.IsGrow = true;
//Sets the appearance of delimiters
mathDelimiter.DelimiterShape = MathDelimiterShapeType.Match;
//Adds the run element for delimiter
IOfficeMathRunElement officeMathRunElement =
mathDelimiter.Equation.Add(0).Functions.Add(0, MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for delimiter equation
(officeMathRunElement.Item as WTextRange).Text = "a+b";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a delimiter equation
IOfficeMathDelimiter mathDelimiter =
officeMath.Functions.Add(0, MathFunctionType.Delimiter) as
IOfficeMathDelimiter;
//Sets the begin character
mathDelimiter.BeginCharacter = "[";
//Sets the end character
mathDelimiter.EndCharacter = "]";
//Enables the flag, to grow delimiter characters to full height of the
arguments
mathDelimiter.IsGrow = true;
//Sets the appearance of delimiters
mathDelimiter.DelimiterShape = MathDelimiterShapeType.Match;
//Adds the run element for delimiter
IOfficeMathRunElement officeMathRunElement =
mathDelimiter.Equation.Add(0).Functions.Add(0, MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for delimiter equation
(officeMathRunElement.Item as WTextRange).Text = "a+b";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing

```

```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN
```

Equation array

You can create a one dimensional array of equations in Word document. The following code example shows how to create an array of equations.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds an equation array
IOfficeMathEquationArray mathEquationArray =
officeMath.Functions.Add(0, MathFunctionType.EquationArray) as
IOfficeMathEquationArray;
//Sets the vertical alignment for equation array
mathEquationArray.VerticalAlignment = MathVerticalAlignment.Center;
//Enables the flag, to distribute the equation array equally within the
container
mathEquationArray.ExpandEquationContainer = true;
//Enables the flag, to expand the equations in an equation array to the
maximum width
mathEquationArray.ExpandEquationContent = true;
//Sets the row spacing rule
mathEquationArray.RowSpacingRule = SpacingRule.Multiple;
//Adds the run element for equation array
IOfficeMathRunElement officeMathRunElement =
mathEquationArray.Equation.Add(0).Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation array
(officeMathRunElement.Item as WTextRange).Text = "x+y+z=0";
//Adds the run element for equation array
officeMathRunElement =
mathEquationArray.Equation.Add(1).Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation array
(officeMathRunElement.Item as WTextRange).Text = "x+y-z=1";
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOOfficeMath = math.MathParagraph.Maths.Add
Dim mathEquationArray As IOOfficeMathEquationArray =
CType(officeMath.Functions.Add(0, MathFunctionType.EquationArray),
IOOfficeMathEquationArray)
'Sets the vertical alignment for equation array
mathEquationArray.VerticalAlignment = MathVerticalAlignment.Center
'Enables the flag, to distribute the equation array equally within the
container
mathEquationArray.ExpandEquationContainer = True
'Enables the flag, to expand the equations in an equation array to the
maximum width
mathEquationArray.ExpandEquationContent = True
'Sets the row spacing rule
mathEquationArray.RowSpacingRule = SpacingRule.Multiple
'Adds the run element for equation array
Dim officeMathRunElement As IOOfficeMathRunElement =
CType(mathEquationArray.Equation.Add(0).Functions.Add(0,
MathFunctionType.RunElement), IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for equation array
CType(officeMathRunElement.Item, WTextRange).Text = "x+y+z=0"
'Adds the run element for equation array
officeMathRunElement =
CType(mathEquationArray.Equation.Add(1).Functions.Add(0,
MathFunctionType.RunElement), IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for equation array
CType(officeMathRunElement.Item, WTextRange).Text = "x+y-z=1"
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds an equation array
IOOfficeMathEquationArray mathEquationArray =
officeMath.Functions.Add(0, MathFunctionType.EquationArray) as
IOOfficeMathEquationArray;
//Sets the vertical alignment for equation array
mathEquationArray.VerticalAlignment = MathVerticalAlignment.Center;

```

```

//Enables the flag, to distribute the equation array equally within the container
mathEquationArray.ExpandEquationContainer = true;
//Enables the flag, to expand the equations in an equation array to the maximum width
mathEquationArray.ExpandEquationContent = true;
//Sets the row spacing rule
mathEquationArray.RowSpacingRule = SpacingRule.Multiple;
//Adds the run element for equation array
IOfficeMathRunElement officeMathRunElement =
mathEquationArray.Equation.Add(0).Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation array
(officeMathRunElement.Item as WTextRange).Text = "x+y+z=0";
//Adds the run element for equation array
officeMathRunElement =
mathEquationArray.Equation.Add(1).Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation array
(officeMathRunElement.Item as WTextRange).Text = "x+y-z=1";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds an equation array
IOfficeMathEquationArray mathEquationArray =
officeMath.Functions.Add(0, MathFunctionType.EquationArray) as
IOfficeMathEquationArray;
//Sets the vertical alignment for equation array
mathEquationArray.VerticalAlignment = MathVerticalAlignment.Center;
//Enables the flag, to distribute the equation array equally within the container
mathEquationArray.ExpandEquationContainer = true;
//Enables the flag, to expand the equations in an equation array to the maximum width
mathEquationArray.ExpandEquationContent = true;
//Sets the row spacing rule
mathEquationArray.RowSpacingRule = SpacingRule.Multiple;

```

```
//Adds the run element for equation array
IOfficeMathRunElement officeMathRunElement =
mathEquationArray.Equation.Add(0).Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation array
(officeMathRunElement.Item as WTextRange).Text = "x+y+z=0";
//Adds the run element for equation array
officeMathRunElement =
mathEquationArray.Equation.Add(1).Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation array
(officeMathRunElement.Item as WTextRange).Text = "x+y-z=1";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds an equation array
IOfficeMathEquationArray mathEquationArray =
officeMath.Functions.Add(0, MathFunctionType.EquationArray) as
IOfficeMathEquationArray;
//Sets the vertical alignment for equation array
mathEquationArray.VerticalAlignment = MathVerticalAlignment.Center;
//Enables the flag, to distribute the equation array equally within the
container
mathEquationArray.ExpandEquationContainer = true;
//Enables the flag, to expand the equations in an equation array to the
maximum width
mathEquationArray.ExpandEquationContent = true;
//Sets the row spacing rule
mathEquationArray.RowSpacingRule = SpacingRule.Multiple;
//Adds the run element for equation array
IOfficeMathRunElement officeMathRunElement =
mathEquationArray.Equation.Add(0).Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation array
(officeMathRunElement.Item as WTextRange).Text = "x+y+z=0";
//Adds the run element for equation array
officeMathRunElement =
```



```

mathEquationArray.Equation.Add(1).Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation array
(officeMathRunElement.Item as WTextRange).Text = "x+y-z=1";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN

```

Fraction

You can create a fraction equation with a numerator and denominator in Word document. The following code example shows how to create a fraction equation.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a fraction equation
IOfficeMathFraction mathFraction =
officeMath.Functions.Add(0, MathFunctionType.Fraction) as
IOfficeMathFraction;
//Sets the denominator for fraction
IOfficeMathRunElement officeMathRunElement =
mathFraction.Numerator.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "a";
//Sets the numerator for fraction
officeMathRunElement =
mathFraction.Denominator.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "b";
//Sets the fraction type
mathFraction.FractionType = MathFractionType.NormalFractionBar;
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOOfficeMath = math.MathParagraph.Maths.Add
Dim mathFraction As IOOfficeMathFraction = CType(officeMath.Functions.Add(0,
MathFunctionType.Fraction), IOOfficeMathFraction)
'Sets the denominator for fraction
Dim officeMathRunElement As IOOfficeMathRunElement =
CType(mathFraction.Denominator.Functions.Add(0, MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
CType(officeMathRunElement.Item, WTextRange).Text = "a"
'Sets the numerator for fraction
officeMathRunElement = CType(mathFraction.Numerator.Functions.Add(0,
MathFunctionType.RunElement), IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
CType(officeMathRunElement.Item, WTextRange).Text = "b"
'Sets the fraction type
mathFraction.FractionType = MathFractionType.NormalFractionBar
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a fraction equation
IOOfficeMathFraction mathFraction =
officeMath.Functions.Add(0, MathFunctionType.Fraction) as
IOOfficeMathFraction;
//Sets the denominator for fraction
IOOfficeMathRunElement officeMathRunElement =
mathFraction.Denominator.Functions.Add(0, MathFunctionType.RunElement) as
IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "a";
//Sets the numerator for fraction
officeMathRunElement =
mathFraction.Numerator.Functions.Add(0, MathFunctionType.RunElement) as
IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "b";

```

```
//Sets the fraction type
mathFraction.FractionType = MathFractionType.NormalFractionBar;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a fraction equation
IOfficeMathFraction mathFraction =
officeMath.Functions.Add(0, MathFunctionType.Fraction) as
IOfficeMathFraction;
//Sets the denominator for fraction
IOfficeMathRunElement officeMathRunElement =
mathFraction.Numerator.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "a";
//Sets the numerator for fraction
officeMathRunElement =
mathFraction.Denominator.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "b";
//Sets the fraction type
mathFraction.FractionType = MathFractionType.NormalFractionBar;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
```

```

WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a fraction equation
IOfficeMathFraction mathFraction =
officeMath.Functions.Add(0, MathFunctionType.Fraction) as
IOfficeMathFraction;
//Sets the denominator for fraction
IOfficeMathRunElement officeMathRunElement =
mathFraction.Numerator.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "a";
//Sets the numerator for fraction
officeMathRunElement =
mathFraction.Denominator.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "b";
//Sets the fraction type
mathFraction.FractionType = MathFractionType.NormalFractionBar;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN

```

Function

You can create trigonometric functions in a Word document. The following code example shows how to create a function.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a function
IOfficeMathFunction mathFunction =
officeMath.Functions.Add(0, MathFunctionType.Function) as
IOfficeMathFunction;
//Sets the function name
IOfficeMathRunElement officeMathRunElement =
mathFunction.FunctionName.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;

```

```

officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "sin";
//Adds the run element for function
officeMathRunElement =
mathFunction.Equation.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for function
(officeMathRunElement.Item as WTextRange).Text = "90";
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOfficeMath = math.MathParagraph.Maths.Add
Dim mathFunction As IOfficeMathFunction = CType(officeMath.Functions.Add(0,
MathFunctionType.Function), IOfficeMathFunction)
'Sets the function name
Dim officeMathRunElement As IOfficeMathRunElement =
CType(mathFunction.FunctionName.Functions.Add(0,
MathFunctionType.RunElement), IOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
CType(officeMathRunElement.Item, WTextRange).Text = "sin"
'Adds the run element for function
officeMathRunElement = CType(mathFunction.Equation.Functions.Add(0,
MathFunctionType.RunElement), IOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for function
CType(officeMathRunElement.Item, WTextRange).Text = "90"
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a function
IOfficeMathFunction mathFunction =

```

```

officeMath.Functions.Add(0, MathFunctionType.Function) as
IOfficeMathFunction;
//Sets the function name
IOfficeMathRunElement officeMathRunElement =
mathFunction.FunctionName.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "sin";
//Adds the run element for function
officeMathRunElement =
mathFunction.Equation.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for function
(officeMathRunElement.Item as WTextRange).Text = "90";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a function
IOfficeMathFunction mathFunction =
officeMath.Functions.Add(0, MathFunctionType.Function) as
IOfficeMathFunction;
//Sets the function name
IOfficeMathRunElement officeMathRunElement =
mathFunction.FunctionName.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "sin";
//Adds the run element for function
officeMathRunElement =
mathFunction.Equation.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for function
(officeMathRunElement.Item as WTextRange).Text = "90";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);

```

```
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a function
IOfficeMathFunction mathFunction =
officeMath.Functions.Add(0, MathFunctionType.Function) as
IOfficeMathFunction;
//Sets the function name
IOfficeMathRunElement officeMathRunElement =
mathFunction.FunctionName.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "sin";
//Adds the run element for function
officeMathRunElement =
mathFunction.Equation.Functions.Add(0, MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for function
(officeMathRunElement.Item as WTextRange).Text = "90";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN
```

Group character

You can group mathematical equations by adding a grouping character at above or below to the corresponding equations. The following code example shows how to create an equation with grouping character.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
```

```

document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a group character equation
IOfficeMathGroupCharacter officeMathGroupCharacter =
officeMath.Functions.Add(0, MathFunctionType.GroupCharacter) as
IOfficeMathGroupCharacter;
//Sets the group character
officeMathGroupCharacter.GroupCharacter = "⌘";
//Enables the flag to align group character at top
officeMathGroupCharacter.HasAlignTop = true;
//Enables the flag to align the text and group character
officeMathGroupCharacter.HasCharacterTop = true;
//Adds the run element for group character
IOfficeMathRunElement officeMathRunElement =
officeMathGroupCharacter.Equation.Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for group character equation
(officeMathRunElement.Item as WTextRange).Text = "a-b";
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOfficeMath = math.MathParagraph.Maths.Add
Dim officeMathGroupCharacter As IOfficeMathGroupCharacter =
CType(officeMath.Functions.Add(0, MathFunctionType.GroupCharacter),
IOfficeMathGroupCharacter)
'Sets the group character
officeMathGroupCharacter.GroupCharacter = "⌘"
'Enables the flag to align group character at top
officeMathGroupCharacter.HasAlignTop = True
'Enables the flag to align the text and group character
officeMathGroupCharacter.HasCharacterTop = True
Dim officeMathRunElement As IOfficeMathRunElement =
CType(officeMathGroupCharacter.Equation.Functions.Add(0,
MathFunctionType.RunElement), IOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for group character equation
CType(officeMathRunElement.Item, WTextRange).Text = "a-b"
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```


UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a group character equation
IOfficeMathGroupCharacter officeMathGroupCharacter =
officeMath.Functions.Add(0, MathFunctionType.GroupCharacter) as
IOfficeMathGroupCharacter;
//Sets the group character
officeMathGroupCharacter.GroupCharacter = "⏟";
//Enables the flag to align group character at top
officeMathGroupCharacter.HasAlignTop = true;
//Enables the flag to align the text and group character
officeMathGroupCharacter.HasCharacterTop = true;
//Adds the run element for group character
IOfficeMathRunElement officeMathRunElement =
officeMathGroupCharacter.Equation.Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for group character equation
(officeMathRunElement.Item as WTextRange).Text = "a-b";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a group character equation
IOfficeMathGroupCharacter officeMathGroupCharacter =
officeMath.Functions.Add(0, MathFunctionType.GroupCharacter) as
IOfficeMathGroupCharacter;
//Sets the group character
officeMathGroupCharacter.GroupCharacter = "⏟";
//Enables the flag to align group character at top

```

```

officeMathGroupCharacter.HasAlignTop = true;
//Enables the flag to align the text and group character
officeMathGroupCharacter.HasCharacterTop = true;
//Adds the run element for group character
IOfficeMathRunElement officeMathRunElement =
officeMathGroupCharacter.Equation.Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for group character equation
(officeMathRunElement.Item as WTextRange).Text = "a-b";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath math = document.LastParagraph.AppendMath();
//Adds a new math
IOfficeMath officeMath = math.MathParagraph.Maths.Add();
//Adds a group character equation
IOfficeMathGroupCharacter officeMathGroupCharacter =
officeMath.Functions.Add(0, MathFunctionType.GroupCharacter) as
IOfficeMathGroupCharacter;
//Sets the group character
officeMathGroupCharacter.GroupCharacter = "⌈";
//Enables the flag to align group character at top
officeMathGroupCharacter.HasAlignTop = true;
//Enables the flag to align the text and group character
officeMathGroupCharacter.HasCharacterTop = true;
//Adds the run element for group character
IOfficeMathRunElement officeMathRunElement =
officeMathGroupCharacter.Equation.Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for group character equation
(officeMathRunElement.Item as WTextRange).Text = "a-b";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform

```

[//https://help.syncfusion.com/file-formats/docio/create-word-document-in-XAMARIN#helper-files-for-XAMARIN](https://help.syncfusion.com/file-formats/docio/create-word-document-in-XAMARIN#helper-files-for-XAMARIN)

Limit

You can add upper limit or lower limit to the mathematical equation. The following code example shows how to create limit equation.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wMath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wMath.MathParagraph.Maths.Add();
//Adds function to the math
IOfficeMathFunction officeMathFunction =
officeMath.Functions.Add(0, MathFunctionType.Function) as
IOfficeMathFunction;
//Adds a mathematical limit equation
IOfficeMathLimit officeMathLimit =
officeMathFunction.FunctionName.Functions.Add(0, MathFunctionType.Limit) as
IOfficeMathLimit;
IOfficeMathRunElement officeMathRunElement =
officeMathLimit.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for limit equation.
(officeMathRunElement.Item as WTextRange).Text = "lim";
//Sets the type of the limit.
officeMathLimit.LimitType = MathLimitType.LowerLimit;
IOfficeMathRunElement officeMathRunElement_limit =
officeMathLimit.Limit.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement_limit.Item = new WTextRange(document);
//Sets the limit value.
(officeMathRunElement_limit.Item as WTextRange).Text = "n=0";
officeMathLimit.LimitType = MathLimitType.LowerLimit;
officeMathRunElement =
officeMathFunction.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for base of the specified equation
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves the Word document
document.Save("Sample.docx");
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
```

```

document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOOfficeMath = math.MathParagraph.Maths.Add
Dim officeMathFunction As IOOfficeMathFunction =
CType(officeMath.Functions.Add(0, MathFunctionType.Function),
IOOfficeMathFunction)
Dim officeMathLimit As IOOfficeMathLimit =
CType(officeMathFunction.FunctionName.Functions.Add(0,
MathFunctionType.Limit), IOOfficeMathLimit)
Dim officeMathRunElement As IOOfficeMathRunElement =
CType(officeMathLimit.Equation.Functions.Add(MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for limit equation.
CType(officeMathRunElement.Item, WTextRange).Text = "lim"
'Sets the type of the limit.
officeMathLimit.LimitType = MathLimitType.LowerLimit
Dim officeMathRunElement_limit As IOOfficeMathRunElement =
CType(officeMathLimit.Limit.Functions.Add(MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement_limit.Item = New WTextRange(document)
'Sets the limit value.
CType(officeMathRunElement_limit.Item, WTextRange).Text = "n=0"
officeMathLimit.LimitType = MathLimitType.LowerLimit
officeMathRunElement =
CType(officeMathFunction.Equation.Functions.Add(MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for base of the specified equation
CType(officeMathRunElement.Item, WTextRange).Text = "x"
'Saves the Word document
document.Save("Sample.docx")
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wMath = document.LastParagraph.AppendMath();
IOOfficeMath officeMath = wMath.MathParagraph.Maths.Add();
//Adds function to the math.
IOOfficeMathFunction officeMathFunction =
officeMath.Functions.Add(0, MathFunctionType.Function) as
IOOfficeMathFunction;
//Adds a mathematical limit equation.
IOOfficeMathLimit officeMathLimit =
officeMathFunction.FunctionName.Functions.Add(0, MathFunctionType.Limit) as
IOOfficeMathLimit;
IOOfficeMathRunElement officeMathRunElement =

```

```

officeMathLimit.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for limit equation.
(officeMathRunElement.Item as WTextRange).Text = "lim";
//Sets the type of the limit.
officeMathLimit.LimitType = MathLimitType.LowerLimit;
IOfficeMathRunElement officeMathRunElement_limit =
officeMathLimit.Limit.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement_limit.Item = new WTextRange(document);
//Sets the limit value.
(officeMathRunElement_limit.Item as WTextRange).Text = "n=0";
officeMathLimit.LimitType = MathLimitType.LowerLimit;
officeMathRunElement =
officeMathFunction.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for base of the specified equation
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wMath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wMath.MathParagraph.Maths.Add();
//Adds function to the math.
IOfficeMathFunction officeMathFunction =
officeMath.Functions.Add(0, MathFunctionType.Function) as
IOfficeMathFunction;
//Adds a mathematical limit equation.
IOfficeMathLimit officeMathLimit =
officeMathFunction.FunctionName.Functions.Add(0, MathFunctionType.Limit) as
IOfficeMathLimit;
IOfficeMathRunElement officeMathRunElement =
officeMathLimit.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for limit equation.
(officeMathRunElement.Item as WTextRange).Text = "lim";
//Sets the type of the limit.
officeMathLimit.LimitType = MathLimitType.LowerLimit;
IOfficeMathRunElement officeMathRunElement_limit =

```

```

officeMathLimit.Limit.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement_limit.Item = new WTextRange(document);
//Sets the limit value.
(officeMathRunElement_limit.Item as WTextRange).Text = "n=0";
officeMathLimit.LimitType = MathLimitType.LowerLimit;
officeMathRunElement =
officeMathFunction.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for base of the specified equation
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wMath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wMath.MathParagraph.Maths.Add();
//Adds function to the math.
IOfficeMathFunction officeMathFunction =
officeMath.Functions.Add(0, MathFunctionType.Function) as
IOfficeMathFunction;
//Adds a mathematical limit equation.
IOfficeMathLimit officeMathLimit =
officeMathFunction.FunctionName.Functions.Add(0, MathFunctionType.Limit) as
IOfficeMathLimit;
IOfficeMathRunElement officeMathRunElement =
officeMathLimit.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for limit equation.
(officeMathRunElement.Item as WTextRange).Text = "lim";
//Sets the type of the limit.
officeMathLimit.LimitType = MathLimitType.LowerLimit;
IOfficeMathRunElement officeMathRunElement_limit =
officeMathLimit.Limit.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement_limit.Item = new WTextRange(document);
//Sets the limit value.
(officeMathRunElement_limit.Item as WTextRange).Text = "n=0";
officeMathLimit.LimitType = MathLimitType.LowerLimit;
officeMathRunElement =
officeMathFunction.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;

```

```

officeMathRunElement.Item = new WTextRange(document);
//Sets text for base of the specified equation
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN

```

Matrix

You can create a matrix equation in a Word document. The following code example shows how to create a matrix equation.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
///Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds matrix equation
IOfficeMathMatrix mathMatrix =
officeMath.Functions.Add(MathFunctionType.Matrix) as IOfficeMathMatrix;
//Sets vertical alignment for matrix
mathMatrix.VerticalAlignment = MathVerticalAlignment.Center;
//Sets width for matrix columns
mathMatrix.ColumnWidth = 1;
//Sets column spacing rule
mathMatrix.ColumnSpacingRule = SpacingRule.OneAndHalf;
//Sets column spacing value
mathMatrix.ColumnSpacing = 3;
//Enables the flag to hide place holders
mathMatrix.HidePlaceHolders = true;
//Sets row spacing rule.
mathMatrix.RowSpacingRule = SpacingRule.Double;
//Sets row spacing value.
mathMatrix.RowSpacing = 2;
//Adds a new column
mathMatrix.Columns.Add();
//Adds a new row
mathMatrix.Rows.Add();
//Sets horizontal alignment for column
mathMatrix.Columns[0].HorizontalAlignment = MathHorizontalAlignment.Left;
//Gets an argument in first cell in first row
officeMath = mathMatrix.Rows[0].Arguments[0];
//Sets text for argument in first cell in first row

```

```

IOfficeMathRunElement officeMathRunElement =
officeMath.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "1";
//Adds a new column
mathMatrix.Columns.Add();
//Adds a new row
mathMatrix.Rows.Add();
//Gets an argument in second cell in first row
officeMath = mathMatrix.Rows[0].Arguments[1];
//Sets text for argument in second cell in first row
officeMathRunElement = officeMath.Functions.Add(MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "2";
//Gets an argument in first cell in second row
officeMath = mathMatrix.Rows[1].Arguments[0];
//Sets text for argument in first cell in second row
officeMathRunElement = officeMath.Functions.Add(MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "3";
//Gets an argument in second cell in second row
officeMath = mathMatrix.Rows[1].Arguments[1];
//Sets text for argument in second cell in second row
officeMathRunElement = officeMath.Functions.Add(MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "4";
//Saves the Word document.
document.Save("Sample.docx");
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOfficeMath = math.MathParagraph.Maths.Add
Dim mathMatrix As IOfficeMathMatrix =
CType(officeMath.Functions.Add(MathFunctionType.Matrix), IOfficeMathMatrix)
'Sets vertical alignment for matrix
mathMatrix.VerticalAlignment = MathVerticalAlignment.Center
'Sets width for matrix columns
mathMatrix.ColumnWidth = 1
'Sets column spacing rule
mathMatrix.ColumnSpacingRule = SpacingRule.OneAndHalf
'Sets column spacing value
mathMatrix.ColumnSpacing = 3
'Enables the flag to hide place holders

```



```

mathMatrix.HidePlaceHolders = True
'Sets row spacing rule.
mathMatrix.RowSpacingRule = SpacingRule.Double
'Sets row spacing value.
mathMatrix.RowSpacing = 2
'Adds a new column
mathMatrix.Columns.Add()
'Adds a new row
mathMatrix.Rows.Add()
'Sets horizontal alignment for column
mathMatrix.Columns(0).HorizontalAlignment = MathHorizontalAlignment.Left
'Gets an argument in first cell in first row
officeMath = mathMatrix.Rows(0).Arguments(0)
'Sets text for argument in first cell in first row
Dim officeMathRunElement As IOOfficeMathRunElement =
CType(officeMath.Functions.Add(MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
CType(officeMathRunElement.Item, WTextRange).Text = "1"
'Adds a new column
mathMatrix.Columns.Add()
'Adds a new row
mathMatrix.Rows.Add()
'Gets an argument in second cell in first row
officeMath = mathMatrix.Rows(0).Arguments(1)
'Sets text for argument in second cell in first row
officeMathRunElement =
CType(officeMath.Functions.Add(MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
CType(officeMathRunElement.Item, WTextRange).Text = "2"
'Gets an argument in first cell in second row
officeMath = mathMatrix.Rows(1).Arguments(0)
'Sets text for argument in first cell in second row
officeMathRunElement =
CType(officeMath.Functions.Add(MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
CType(officeMathRunElement.Item, WTextRange).Text = "3"
'Gets an argument in second cell in second row
officeMath = mathMatrix.Rows(1).Arguments(1)
'Sets text for argument in second cell in second row
officeMathRunElement =
CType(officeMath.Functions.Add(MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
CType(officeMathRunElement.Item, WTextRange).Text = "4"
'Saves the Word document.
document.Save("Sample.docx")
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();

```

```

///Adds one section and one paragraph to the document
document.EnsureMinimal();
///Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
///Adds matrix equation
IOfficeMathMatrix mathMatrix =
officeMath.Functions.Add(MathFunctionType.Matrix) as IOfficeMathMatrix;
///Sets vertical alignment for matrix
mathMatrix.VerticalAlignment = MathVerticalAlignment.Center;
///Sets width for matrix columns
mathMatrix.ColumnWidth = 1;
///Sets column spacing rule
mathMatrix.ColumnSpacingRule = SpacingRule.OneAndHalf;
///Sets column spacing value
mathMatrix.ColumnSpacing = 3;
///Enables the flag to hide place holders
mathMatrix.HidePlaceHolders = true;
///Sets row spacing rule.
mathMatrix.RowSpacingRule = SpacingRule.Double;
///Sets row spacing value.
mathMatrix.RowSpacing = 2;
///Adds a new column
mathMatrix.Columns.Add();
///Adds a new row
mathMatrix.Rows.Add();
///Sets horizontal alignment for column
mathMatrix.Columns[0].HorizontalAlignment = MathHorizontalAlignment.Left;
///Gets an argument in first cell in first row
officeMath = mathMatrix.Rows[0].Arguments[0];
///Sets text for argument in first cell in first row
IOfficeMathRunElement officeMathRunElement =
officeMath.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "1";
///Adds a new column
mathMatrix.Columns.Add();
///Adds a new row
mathMatrix.Rows.Add();
///Gets an argument in second cell in first row
officeMath = mathMatrix.Rows[0].Arguments[1];
///Sets text for argument in second cell in first row
officeMathRunElement = officeMath.Functions.Add(MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "2";
///Gets an argument in first cell in second row
officeMath = mathMatrix.Rows[1].Arguments[0];
///Sets text for argument in first cell in second row
officeMathRunElement = officeMath.Functions.Add(MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "3";
///Gets an argument in second cell in second row
officeMath = mathMatrix.Rows[1].Arguments[1];
///Sets text for argument in second cell in second row

```

```

officeMathRunElement = officeMath.Functions.Add(MathFunctionType.RunElement)
as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "4";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
///Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds matrix equation
IOOfficeMathMatrix mathMatrix =
officeMath.Functions.Add(MathFunctionType.Matrix) as IOOfficeMathMatrix;
//Sets vertical alignment for matrix
mathMatrix.VerticalAlignment = MathVerticalAlignment.Center;
//Sets width for matrix columns
mathMatrix.ColumnWidth = 1;
//Sets column spacing rule
mathMatrix.ColumnSpacingRule = SpacingRule.OneAndHalf;
//Sets column spacing value
mathMatrix.ColumnSpacing = 3;
//Enables the flag to hide place holders
mathMatrix.HidePlaceHolders = true;
//Sets row spacing rule.
mathMatrix.RowSpacingRule = SpacingRule.Double;
//Sets row spacing value.
mathMatrix.RowSpacing = 2;
//Adds a new column
mathMatrix.Columns.Add();
//Adds a new row
mathMatrix.Rows.Add();
//Sets horizontal alignment for column
mathMatrix.Columns[0].HorizontalAlignment = MathHorizontalAlignment.Left;
//Gets an argument in first cell in first row
officeMath = mathMatrix.Rows[0].Arguments[0];
//Sets text for argument in first cell in first row
IOOfficeMathRunElement officeMathRunElement =
officeMath.Functions.Add(MathFunctionType.RunElement) as
IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "1";
//Adds a new column
mathMatrix.Columns.Add();

```

```
//Adds a new row
mathMatrix.Rows.Add();
//Gets an argument in second cell in first row
officeMath = mathMatrix.Rows[0].Arguments[1];
//Sets text for argument in second cell in first row
officeMathRunElement = officeMath.Functions.Add(MathFunctionType.RunElement)
as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "2";
//Gets an argument in first cell in second row
officeMath = mathMatrix.Rows[1].Arguments[0];
//Sets text for argument in first cell in second row
officeMathRunElement = officeMath.Functions.Add(MathFunctionType.RunElement)
as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "3";
//Gets an argument in second cell in second row
officeMath = mathMatrix.Rows[1].Arguments[1];
//Sets text for argument in second cell in second row
officeMathRunElement = officeMath.Functions.Add(MathFunctionType.RunElement)
as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "4";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds matrix equation
IOOfficeMathMatrix mathMatrix =
officeMath.Functions.Add(MathFunctionType.Matrix) as IOOfficeMathMatrix;
//Sets vertical alignment for matrix
mathMatrix.VerticalAlignment = MathVerticalAlignment.Center;
//Sets width for matrix columns
mathMatrix.ColumnWidth = 1;
//Sets column spacing rule
mathMatrix.ColumnSpacingRule = SpacingRule.OneAndHalf;
//Sets column spacing value
mathMatrix.ColumnSpacing = 3;
//Enables the flag to hide place holders
mathMatrix.HidePlaceHolders = true;
//Sets row spacing rule.
mathMatrix.RowSpacingRule = SpacingRule.Double;
```

```

//Sets row spacing value.
mathMatrix.RowSpacing = 2;
//Adds a new column
mathMatrix.Columns.Add();
//Adds a new row
mathMatrix.Rows.Add();
//Sets horizontal alignment for column
mathMatrix.Columns[0].HorizontalAlignment = MathHorizontalAlignment.Left;
//Gets an argument in first cell in first row
officeMath = mathMatrix.Rows[0].Arguments[0];
//Sets text for argument in first cell in first row
IOfficeMathRunElement officeMathRunElement =
officeMath.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "1";
//Adds a new column
mathMatrix.Columns.Add();
//Adds a new row
mathMatrix.Rows.Add();
//Gets an argument in second cell in first row
officeMath = mathMatrix.Rows[0].Arguments[1];
//Sets text for argument in second cell in first row
officeMathRunElement = officeMath.Functions.Add(MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "2";
//Gets an argument in first cell in second row
officeMath = mathMatrix.Rows[1].Arguments[0];
//Sets text for argument in first cell in second row
officeMathRunElement = officeMath.Functions.Add(MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "3";
//Gets an argument in second cell in second row
officeMath = mathMatrix.Rows[1].Arguments[1];
//Sets text for argument in second cell in second row
officeMathRunElement = officeMath.Functions.Add(MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "4";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN

```

N-Array

You can create an equation with common large operators such as summation, integrals, union, intersection, logical OR, logical AND, products and co-products. The following code example shows how to create a summation with limits.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wMath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wMath.MathParagraph.Maths.Add();
//Adds a N-Array equation.
IOfficeMathNArray officeMathNArray = officeMath.Functions.Add(0,
MathFunctionType.NArray) as IOfficeMathNArray;
//Sets N-Array character.
officeMathNArray.NArrayCharacter = "Σ";
//Enables the flag, to grow N-array character to full height of the
arguments
officeMathNArray.HasGrow = false;
//Enables the flag to hide lower limit
officeMathNArray.HideLowerLimit = false;
//Enables the flag to hide upper limit
officeMathNArray.HideUpperLimit = false;
//Sets false to set limit position on above the summation
officeMathNArray.SubSuperscriptLimit = false;
IOfficeMathRunElement officeMathRunElement =
officeMathNArray.Subscript.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for superscript property of NArray equation.
(officeMathRunElement.Item as WTextRange).Text = "n=1";
officeMathRunElement =
officeMathNArray.Superscript.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "10";
officeMathRunElement =
officeMathNArray.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for NArray equation.
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves the Word document.
document.Save("Sample.docx");
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
```

```

'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOOfficeMath = math.MathParagraph.Maths.Add
Dim officeMathNArray As IOOfficeMathNArray =
CType(officeMath.Functions.Add(0, MathFunctionType.NArray),
IOOfficeMathNArray)
'Sets N-Array character.
officeMathNArray.NArrayCharacter = "_"
'Enables the flag, to grow N-array character to full height of the arguments
officeMathNArray.HasGrow = False
'Enables the flag to hide lower limit
officeMathNArray.HideLowerLimit = False
'Enables the flag to hide upper limit
officeMathNArray.HideUpperLimit = False
'Enables the flag to set limit position as SubSuperscript
officeMathNArray.SubSuperscriptLimit = True
Dim officeMathRunElement As IOOfficeMathRunElement =
CType(officeMathNArray.Subscript.Functions.Add(MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for superscript property of NArray equation.
CType(officeMathRunElement.Item, WTextRange).Text = "n=1"
officeMathRunElement =
CType(officeMathNArray.Superscript.Functions.Add(MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
CType(officeMathRunElement.Item, WTextRange).Text = "10"
officeMathRunElement =
CType(officeMathNArray.Equation.Functions.Add(MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for NArray equation.
CType(officeMathRunElement.Item, WTextRange).Text = "x"
'Saves the Word document.
document.Save("Sample.docx")
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wMath = document.LastParagraph.AppendMath();
IOOfficeMath officeMath = wMath.MathParagraph.Maths.Add();
//Adds a N-Array equation.
IOOfficeMathNArray officeMathNArray = officeMath.Functions.Add(0,
MathFunctionType.NArray) as IOOfficeMathNArray;
//Sets N-Array character.
officeMathNArray.NArrayCharacter = "Σ";
//Enables the flag, to grow N-array character to full height of the arguments
officeMathNArray.HasGrow = false;

```

```

//Enables the flag to hide lower limit
officeMathNArray.HideLowerLimit = false;
//Enables the flag to hide upper limit
officeMathNArray.HideUpperLimit = false;
//Enables the flag to set limit position as SubSuperscript
officeMathNArray.SubSuperscriptLimit = true;
IOfficeMathRunElement officeMathRunElement =
officeMathNArray.Subscript.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for superscript property of NArray equation.
(officeMathRunElement.Item as WTextRange).Text = "n=1";
officeMathRunElement =
officeMathNArray.Superscript.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "10";
officeMathRunElement =
officeMathNArray.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for NArray equation.
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wMath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wMath.MathParagraph.Maths.Add();
//Adds a N-Array equation.
IOfficeMathNArray officeMathNArray = officeMath.Functions.Add(0,
MathFunctionType.NArray) as IOfficeMathNArray;
//Sets N-Array character.
officeMathNArray.NArrayCharacter = "Σ";
//Enables the flag, to grow N-array character to full height of the
arguments
officeMathNArray.HasGrow = false;
//Enables the flag to hide lower limit
officeMathNArray.HideLowerLimit = false;
//Enables the flag to hide upper limit
officeMathNArray.HideUpperLimit = false;
//Enables the flag to set limit position as SubSuperscript
officeMathNArray.SubSuperscriptLimit = true;

```



```

IOfficeMathRunElement officeMathRunElement =
officeMathNArray.Subscript.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for superscript property of NArray equation.
(officeMathRunElement.Item as WTextRange).Text = "n=1";
officeMathRunElement =
officeMathNArray.Superscript.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "10";
officeMathRunElement =
officeMathNArray.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for NArray equation.
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wMath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wMath.MathParagraph.Maths.Add();
//Adds a N-Array equation.
IOfficeMathNArray officeMathNArray = officeMath.Functions.Add(0,
MathFunctionType.NArray) as IOfficeMathNArray;
//Sets N-Array character.
officeMathNArray.NArrayCharacter = "Σ";
//Enables the flag, to grow N-array character to full height of the
arguments
officeMathNArray.HasGrow = false;
//Enables the flag to hide lower limit
officeMathNArray.HideLowerLimit = false;
//Enables the flag to hide upper limit
officeMathNArray.HideUpperLimit = false;
//Enables the flag to set limit position as SubSuperscript
officeMathNArray.SubSuperscriptLimit = true;
IOfficeMathRunElement officeMathRunElement =
officeMathNArray.Subscript.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for superscript property of NArray equation.
(officeMathRunElement.Item as WTextRange).Text = "n=1";
officeMathRunElement =

```

```

officeMathNArray.Superscript.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "10";
officeMathRunElement =
officeMathNArray.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for NArray equation.
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN

```

Radical

You can create a radical equation in Word document. The following example shows how to create a radical equation.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a radical equation
IOfficeMathRadical officeMathRadical = officeMath.Functions.Add(0,
MathFunctionType.Radical) as IOfficeMathRadical;
//Sets false to show degree in radical
officeMathRadical.HideDegree = false;
//Adds a degree for radical equation
IOfficeMathRunElement officeMathRunElement =
officeMathRadical.Degree.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "2";
//Adds an run element for radical
officeMathRunElement =
officeMathRadical.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets the radicand text for radical equation.
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);

```

```
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOOfficeMath = math.MathParagraph.Maths.Add
Dim officeMathRadical As IOOfficeMathRadical =
CType(officeMath.Functions.Add(0, MathFunctionType.Radical),
IOOfficeMathRadical)
'Sets false to show degree in radical
officeMathRadical.HideDegree = False
'Adds a degree for radical equation
Dim officeMathRunElement As IOOfficeMathRunElement =
CType(officeMathRadical.Degree.Functions.Add(MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
CType(officeMathRunElement.Item, WTextRange).Text = "2"
'Adds an equation for radical
officeMathRunElement =
CType(officeMathRadical.Equation.Functions.Add(MathFunctionType.RunElement),
IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets the text for radical equation.
CType(officeMathRunElement.Item, WTextRange).Text = "x"
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a radical equation
IOOfficeMathRadical officeMathRadical = officeMath.Functions.Add(0,
MathFunctionType.Radical) as IOOfficeMathRadical;
//Sets false to show degree in radical
officeMathRadical.HideDegree = false;
//Adds a degree for radical equation
IOOfficeMathRunElement officeMathRunElement =
officeMathRadical.Degree.Functions.Add(MathFunctionType.RunElement) as
IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "2";
```

```
//Adds an equation for radical
officeMathRunElement =
officeMathRadical.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets the text for radical equation.
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a radical equation
IOfficeMathRadical officeMathRadical = officeMath.Functions.Add(0,
MathFunctionType.Radical) as IOfficeMathRadical;
//Sets false to show degree in radical
officeMathRadical.HideDegree = false;
//Adds a degree for radical equation
IOfficeMathRunElement officeMathRunElement =
officeMathRadical.Degree.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "2";
//Adds an equation for radical
officeMathRunElement =
officeMathRadical.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets the text for radical equation.
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Creates a new Word document
```

```

WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a radical equation
IOfficeMathRadical officeMathRadical = officeMath.Functions.Add(0,
MathFunctionType.Radical) as IOfficeMathRadical;
//Sets false to show degree in radical
officeMathRadical.HideDegree = false;
//Adds a degree for radical equation
IOfficeMathRunElement officeMathRunElement =
officeMathRadical.Degree.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "2";
//Adds an equation for radical
officeMathRunElement =
officeMathRadical.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets the text for radical equation.
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN

```

Phantom

You can create a phantom equation to add the spacing of the phantom

without displaying that base and suppressing part of the glyph from spacing considerations. The following code example shows how to create a phantom equation.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a radical equation
IOfficeMathRadical officeMathRadical = officeMath.Functions.Add(0,
MathFunctionType.Radical) as IOfficeMathRadical;
//Adds a degree for radical

```

```

IOfficeMathRunElement officeMathRunElement =
officeMathRadical.Degree.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "2";
//Adds a phantom equation in radical.
IOfficeMathPhantom officeMathPhantom =
officeMathRadical.Equation.Functions.Add(0, MathFunctionType.Phantom) as
IOfficeMathPhantom;
//Enables the flag, to show the contents of phantom
officeMathPhantom.Show = true;
//Enables the flag, to transparent the phantom
officeMathPhantom.Transparent = true;
//Enables the flag, to ignore the ascent of the phantom contents in spacing
officeMathPhantom.ZeroAscent = true;
//Enables the flag, to ignore the descent of the phantom contents in spacing
officeMathPhantom.ZeroDescent = true;
//Enables the flag, to ignore the width of a phantom contents in spacing
officeMathPhantom.ZeroWidth = true;
//Adds a run element for phantom
officeMathRunElement =
officeMathPhantom.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for phantom equation
(officeMathRunElement.Item as WTextRange).Text = "a-b";
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOfficeMath = math.MathParagraph.Maths.Add
Dim officeMathRadical As IOfficeMathRadical =
CType(officeMath.Functions.Add(0, MathFunctionType.Radical),
IOfficeMathRadical)
Dim officeMathRunElement As IOfficeMathRunElement =
CType(officeMathRadical.Degree.Functions.Add(MathFunctionType.RunElement),
IOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
CType(officeMathRunElement.Item, WTextRange).Text = "2"
Dim officeMathPhantom As IOfficeMathPhantom =
CType(officeMathRadical.Equation.Functions.Add(0, MathFunctionType.Phantom),
IOfficeMathPhantom)
'Enables the flag, to show the contents of phantom
officeMathPhantom.Show = True
'Enables the flag, to transparent the phantom
officeMathPhantom.Transparent = True

```

```

'Enables the flag, to ignore the ascent of the phantom contents in spacing
officeMathPhantom.ZeroAscent = True
'Enables the flag, to ignore the descent of the phantom contents in spacing
officeMathPhantom.ZeroDescent = True
'Enables the flag, to ignore the width of a phantom contents in spacing
officeMathPhantom.ZeroWidth = True
'Adds a run element for math phantom
officeMathRunElement =
CType (officeMathPhantom.Equation.Functions.Add(MathFunctionType.RunElement),
IOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for phantom equation
CType (officeMathRunElement.Item, WTextRange).Text = "a-b"
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a radical equation
IOfficeMathRadical officeMathRadical = officeMath.Functions.Add(0,
MathFunctionType.Radical) as IOfficeMathRadical;
IOfficeMathRunElement officeMathRunElement =
officeMathRadical.Degree.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "2";
//Adds a phantom equation in radical.
IOfficeMathPhantom officeMathPhantom =
officeMathRadical.Equation.Functions.Add(0, MathFunctionType.Pantom) as
IOfficeMathPhantom;
//Enables the flag, to show the contents of phantom
officeMathPhantom.Show = true;
//Enables the flag, to transparent the phantom
officeMathPhantom.Transparent = true;
//Enables the flag, to ignore the ascent of the phantom contents in spacing
officeMathPhantom.ZeroAscent = true;
//Enables the flag, to ignore the descent of the phantom contents in spacing
officeMathPhantom.ZeroDescent = true;
//Enables the flag, to ignore the width of a phantom contents in spacing
officeMathPhantom.ZeroWidth = true;
//Adds a run element for math phantom
officeMathRunElement =
officeMathPhantom.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for phantom equation
(officeMathRunElement.Item as WTextRange).Text = "a-b";

```

```
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a radical equation
IOfficeMathRadical officeMathRadical = officeMath.Functions.Add(0,
MathFunctionType.Radical) as IOfficeMathRadical;
IOfficeMathRunElement officeMathRunElement =
officeMathRadical.Degree.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "2";
//Adds a phantom equation in radical.
IOfficeMathPhantom officeMathPhantom =
officeMathRadical.Equation.Functions.Add(0, MathFunctionType.Pantom) as
IOfficeMathPhantom;
//Enables the flag, to show the contents of phantom
officeMathPhantom.Show = true;
//Enables the flag, to transparent the phantom
officeMathPhantom.Transparent = true;
//Enables the flag, to ignore the ascent of the phantom contents in spacing
officeMathPhantom.ZeroAscent = true;
//Enables the flag, to ignore the descent of the phantom contents in spacing
officeMathPhantom.ZeroDescent = true;
//Enables the flag, to ignore the width of a phantom contents in spacing
officeMathPhantom.ZeroWidth = true;
//Adds a run element for math phantom
officeMathRunElement =
officeMathPhantom.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for phantom equation
(officeMathRunElement.Item as WTextRange).Text = "a-b";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```


XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a radical equation
IOfficeMathRadical officeMathRadical = officeMath.Functions.Add(0,
MathFunctionType.Radical) as IOfficeMathRadical;
IOfficeMathRunElement officeMathRunElement =
officeMathRadical.Degree.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
(officeMathRunElement.Item as WTextRange).Text = "2";
//Adds a phantom equation in radical.
IOfficeMathPhantom officeMathPhantom =
officeMathRadical.Equation.Functions.Add(0, MathFunctionType.Pantom) as
IOfficeMathPhantom;
//Enables the flag, to show the contents of phantom
officeMathPhantom.Show = true;
//Enables the flag, to transparent the phantom
officeMathPhantom.Transparent = true;
//Enables the flag, to ignore the ascent of the phantom contents in spacing
officeMathPhantom.ZeroAscent = true;
//Enables the flag, to ignore the descent of the phantom contents in spacing
officeMathPhantom.ZeroDescent = true;
//Enables the flag, to ignore the width of a phantom contents in spacing
officeMathPhantom.ZeroWidth = true;
//Adds a run element for math phantom
officeMathRunElement =
officeMathPhantom.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for phantom equation
(officeMathRunElement.Item as WTextRange).Text = "a-b";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN

```

SubSuperscript

You can add a superscript or subscript equation in a Word document. The following code shows how to create a superscript equation.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a superscript equation
IOfficeMathScript officeMathScript = officeMath.Functions.Add(0,
MathFunctionType.SubSuperscript) as IOfficeMathScript;
//Sets the type of the script as superscript.
officeMathScript.ScriptType = MathScriptType.Superscript;
//Adds a run element for superscript.
IOfficeMathRunElement officeMathRunElement =
officeMathScript.Script.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
WTextRange textRange = officeMathRunElement.Item as WTextRange;
//Sets text for superscript.
textRange.Text = "2";
//Adds run element for equation
officeMathRunElement =
officeMathScript.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
Dim officeMath As IOfficeMath = math.MathParagraph.Maths.Add
Dim officeMathScript As IOfficeMathScript =
CType(officeMath.Functions.Add(0, MathFunctionType.SubSuperscript),
IOfficeMathScript)
'Sets the type of the script.
officeMathScript.ScriptType = MathScriptType.Superscript
'Adds a run element for script.
Dim officeMathRunElement As IOfficeMathRunElement =
CType(officeMathScript.Script.Functions.Add(MathFunctionType.RunElement),
IOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
Dim textRange As WTextRange = CType(officeMathRunElement.Item, WTextRange)
'Sets text for script.

```

```

textRange.Text = "2"
'Adds run element for equation
officeMathRunElement =
CType (officeMathScript.Equation.Functions.Add (MathFunctionType.RunElement),
IOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange (document)
'Sets text
CType (officeMathRunElement.Item, WTextRange).Text = "x"
'Saves the Word document
document.Save ("Sample.docx", FormatType.Docx)
'Closes the document
document.Close ()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument ();
//Adds one section and one paragraph to the document
document.EnsureMinimal ();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath ();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add ();
//Adds a subsuperscript equation
IOfficeMathScript officeMathScript = officeMath.Functions.Add (0,
MathFunctionType.SubSuperscript) as IOfficeMathScript;
//Sets the type of the script.
officeMathScript.ScriptType = MathScriptType.Superscript;
//Adds a run element for script.
IOfficeMathRunElement officeMathRunElement =
officeMathScript.Script.Functions.Add (MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange (document);
WTextRange textRange = officeMathRunElement.Item as WTextRange;
//Sets text for script.
textRange.Text = "2";
//Adds run element for equation
officeMathRunElement =
officeMathScript.Equation.Functions.Add (MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange (document);
//Sets text
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream ();
//Saves the Word document to MemoryStream
await document.SaveAsync (stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save (stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument ();

```

```

//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a subsuperscript equation
IOfficeMathScript officeMathScript = officeMath.Functions.Add(0,
MathFunctionType.SubSuperscript) as IOfficeMathScript;
//Sets the type of the script.
officeMathScript.ScriptType = MathScriptType.Superscript;
//Adds a run element for script.
IOfficeMathRunElement officeMathRunElement =
officeMathScript.Script.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
WTextRange textRange = officeMathRunElement.Item as WTextRange;
//Sets text for script.
textRange.Text = "2";
//Adds run element for equation
officeMathRunElement =
officeMathScript.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a subsuperscript equation
IOfficeMathScript officeMathScript = officeMath.Functions.Add(0,
MathFunctionType.SubSuperscript) as IOfficeMathScript;
//Sets the type of the script.
officeMathScript.ScriptType = MathScriptType.Superscript;
//Adds a run element for script.
IOfficeMathRunElement officeMathRunElement =
officeMathScript.Script.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
WTextRange textRange = officeMathRunElement.Item as WTextRange;
//Sets text for script.
textRange.Text = "2";

```

```
//Adds run element for equation
officeMathRunElement =
officeMathScript.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text
(officeMathRunElement.Item as WTextRange).Text = "x";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN
```

Left SubSuperscript

You can add superscript and subscript on the left side of mathematical equation. The following code example shows how to add superscript and subscript on the left side of the equation.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a left subsuperscript equation.
IOfficeMathLeftScript officeMathLeftSubScript = officeMath.Functions.Add(0,
MathFunctionType.LeftSubSuperscript) as IOfficeMathLeftScript;
//Adds run element for left subscript
IOfficeMathRunElement officeMathRunElement =
officeMathLeftSubScript.Subscript.Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for subscript
(officeMathRunElement.Item as WTextRange).Text = "1";
//Adds a run element for left superscript
officeMathRunElement =
officeMathLeftSubScript.Superscript.Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for left superscript.
(officeMathRunElement.Item as WTextRange).Text = "n";
officeMathRunElement =
officeMathLeftSubScript.Equation.Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation.
(officeMathRunElement.Item as WTextRange).Text = "Y";
```

```
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
Dim officeMath As IOOfficeMath = math.MathParagraph.Maths.Add
'Adds a left subsuperscript equation.
Dim officeMathLeftSubScript As IOOfficeMathLeftScript =
CType(officeMath.Functions.Add(0, MathFunctionType.LeftSubSuperscript),
IOOfficeMathLeftScript)
'Adds run element for left subsuperscript
Dim officeMathRunElement As IOOfficeMathRunElement =
CType(officeMathLeftSubScript.Subscript.Functions.Add(0,
MathFunctionType.RunElement), IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
CType(officeMathRunElement.Item, WTextRange).Text = "1"
officeMathRunElement =
CType(officeMathLeftSubScript.Superscript.Functions.Add(0,
MathFunctionType.RunElement), IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'//Sets text for left superscript.
CType(officeMathRunElement.Item, WTextRange).Text = "n"
officeMathRunElement =
CType(officeMathLeftSubScript.Equation.Functions.Add(0,
MathFunctionType.RunElement), IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for left equation.
CType(officeMathRunElement.Item, WTextRange).Text = "Y"
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()
```

UWP

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a left subsuperscript equation.
IOOfficeMathLeftScript officeMathLeftSubScript = officeMath.Functions.Add(0,
MathFunctionType.LeftSubSuperscript) as IOOfficeMathLeftScript;
//Adds run element for left subscript
IOOfficeMathRunElement officeMathRunElement =
```

```

officeMathLeftSubScript.Subscript.Functions.Add(0,
MathFunctionType.RunElement) as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for subscript
(officeMathRunElement.Item as WTextRange).Text = "1";
//Adds a run element for left superscript
officeMathRunElement =
officeMathLeftSubScript.Superscript.Functions.Add(0,
MathFunctionType.RunElement) as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for left superscript.
(officeMathRunElement.Item as WTextRange).Text = "n";
officeMathRunElement =
officeMathLeftSubScript.Equation.Functions.Add(0,
MathFunctionType.RunElement) as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation.
(officeMathRunElement.Item as WTextRange).Text = "Y";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a left subsuperscript equation.
IOOfficeMathLeftScript officeMathLeftSubScript = officeMath.Functions.Add(0,
MathFunctionType.LeftSubSuperscript) as IOOfficeMathLeftScript;
//Adds run element for left subscript
IOOfficeMathRunElement officeMathRunElement =
officeMathLeftSubScript.Subscript.Functions.Add(0,
MathFunctionType.RunElement) as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for subscript
(officeMathRunElement.Item as WTextRange).Text = "1";
//Adds a run element for left superscript
officeMathRunElement =
officeMathLeftSubScript.Superscript.Functions.Add(0,
MathFunctionType.RunElement) as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for left superscript.
(officeMathRunElement.Item as WTextRange).Text = "n";
officeMathRunElement =

```

```

officeMathLeftSubScript.Equation.Functions.Add(0,
MathFunctionType.RunElement) as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation.
(officeMathRunElement.Item as WTextRange).Text = "Y";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a left subsuperscript equation.
IOOfficeMathLeftScript officeMathLeftSubScript = officeMath.Functions.Add(0,
MathFunctionType.LeftSubSuperscript) as IOOfficeMathLeftScript;
//Adds run element for left subscript
IOOfficeMathRunElement officeMathRunElement =
officeMathLeftSubScript.Subscript.Functions.Add(0,
MathFunctionType.RunElement) as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for subscript
(officeMathRunElement.Item as WTextRange).Text = "1";
//Adds a run element for left superscript
officeMathRunElement =
officeMathLeftSubScript.Superscript.Functions.Add(0,
MathFunctionType.RunElement) as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for left superscript.
(officeMathRunElement.Item as WTextRange).Text = "n";
officeMathRunElement =
officeMathLeftSubScript.Equation.Functions.Add(0,
MathFunctionType.RunElement) as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation.
(officeMathRunElement.Item as WTextRange).Text = "Y";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform

```



```
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-XAMARIN#helper-files-for-XAMARIN
```

Right SubSuperscript

You can add superscript and subscript on the right side of mathematical equation. The following code example shows how to add superscript and subscript on the right side of the equation.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a right subsuperscript equation
IOfficeMathRightScript officeMathRightScript = officeMath.Functions.Add(0,
MathFunctionType.RightSubSuperscript) as IOfficeMathRightScript;
//Sets false to align subscript and superscript horizontally
officeMathRightScript.IsSkipAlign = true;
//Adds run element for right subscript.
IOfficeMathRunElement officeMathRunElement =
officeMathRightScript.Subscript.Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for right subscript
(officeMathRunElement.Item as WTextRange).Text = "1";
//Adds run element for right superscript
officeMathRunElement =
officeMathRightScript.Superscript.Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for right superscript.
(officeMathRunElement.Item as WTextRange).Text = "n";
officeMathRunElement =
officeMathRightScript.Equation.Functions.Add(0, MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation.
(officeMathRunElement.Item as WTextRange).Text = "Y";
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As WordDocument = New WordDocument
'Adds one section and one paragraph to the document
document.EnsureMinimal()
'Appends a new mathematical equation to the paragraph
Dim math As WMath = document.LastParagraph.AppendMath
'Adds a new math
```

```

Dim officeMath As IOOfficeMath = math.MathParagraph.Maths.Add
Dim officeMathRightScript As IOOfficeMathRightScript =
CType(officeMath.Functions.Add(0, MathFunctionType.RightSubSuperscript),
IOOfficeMathRightScript)
'Sets false to align subscripts and superscripts horizontally
officeMathRightScript.IsSkipAlign = True
'Adds run element for right subscript equation.
Dim officeMathRunElement As IOOfficeMathRunElement =
CType(officeMathRightScript.Subscript.Functions.Add(0,
MathFunctionType.RunElement), IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for subscript.
CType(officeMathRunElement.Item, WTextRange).Text = "1"
'Adds run element for right superscript equation.
officeMathRunElement =
CType(officeMathRightScript.Superscript.Functions.Add(0,
MathFunctionType.RunElement), IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for superscript.
CType(officeMathRunElement.Item, WTextRange).Text = "n"
officeMathRunElement = CType(officeMathRightScript.Equation.Functions.Add(0,
MathFunctionType.RunElement), IOOfficeMathRunElement)
officeMathRunElement.Item = New WTextRange(document)
'Sets text for superscript.
CType(officeMathRunElement.Item, WTextRange).Text = "Y"
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a right superscript equation
IOOfficeMathRightScript officeMathRightScript = officeMath.Functions.Add(0,
MathFunctionType.RightSubSuperscript) as IOOfficeMathRightScript;
//Sets false to align subscript and superscript horizontally
officeMathRightScript.IsSkipAlign = true;
//Adds run element for right subscript.
IOOfficeMathRunElement officeMathRunElement =
officeMathRightScript.Subscript.Functions.Add(0,
MathFunctionType.RunElement) as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for right subscript
(officeMathRunElement.Item as WTextRange).Text = "1";
//Adds run element for right superscript
officeMathRunElement =
officeMathRightScript.Superscript.Functions.Add(0,
MathFunctionType.RunElement) as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);

```

```
//Sets text for right superscript.
(officeMathRunElement.Item as WTextRange).Text = "n";
officeMathRunElement =
officeMathRightScript.Equation.Functions.Add(0, MathFunctionType.RunElement)
as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation.
(officeMathRunElement.Item as WTextRange).Text = "Y";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a right subsuperscript equation
IOOfficeMathRightScript officeMathRightScript = officeMath.Functions.Add(0,
MathFunctionType.RightSubSuperscript) as IOOfficeMathRightScript;
//Sets false to align subscript and superscript horizontally
officeMathRightScript.IsSkipAlign = true;
//Adds run element for right subscript.
IOOfficeMathRunElement officeMathRunElement =
officeMathRightScript.Subscript.Functions.Add(0,
MathFunctionType.RunElement) as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for right subscript
(officeMathRunElement.Item as WTextRange).Text = "1";
//Adds run element for right superscript
officeMathRunElement =
officeMathRightScript.Superscript.Functions.Add(0,
MathFunctionType.RunElement) as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for right superscript.
(officeMathRunElement.Item as WTextRange).Text = "n";
officeMathRunElement =
officeMathRightScript.Equation.Functions.Add(0, MathFunctionType.RunElement)
as IOOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation.
(officeMathRunElement.Item as WTextRange).Text = "Y";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
```

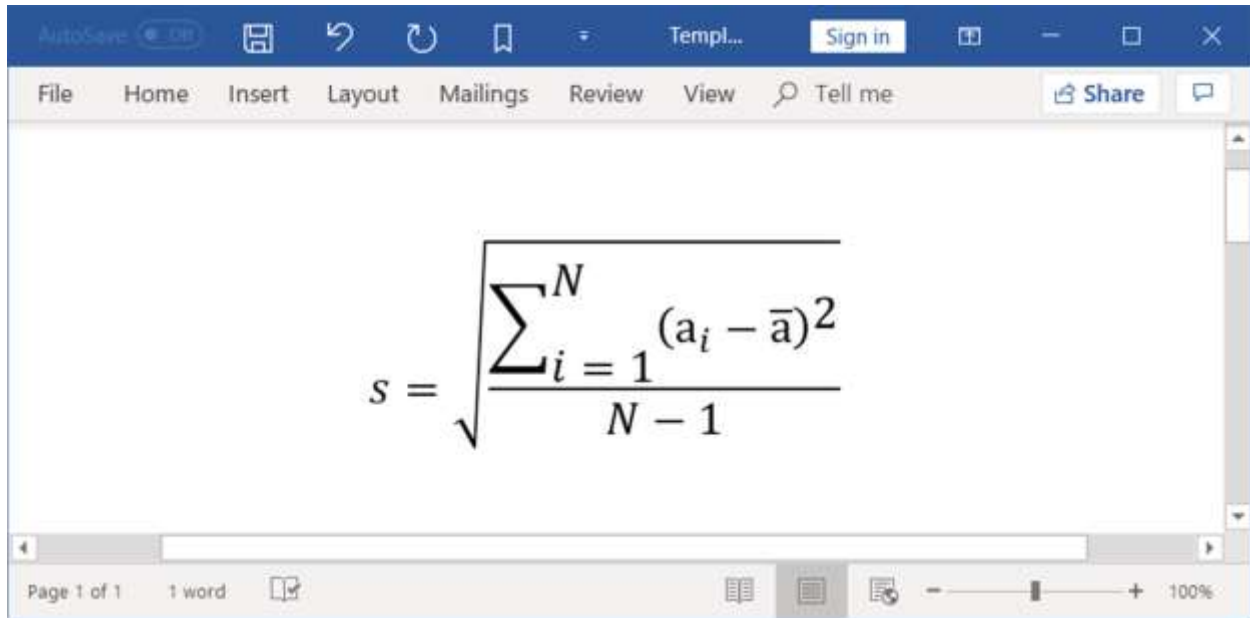
```
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds one section and one paragraph to the document
document.EnsureMinimal();
//Appends a new mathematical equation to the paragraph
WMath wmath = document.LastParagraph.AppendMath();
IOfficeMath officeMath = wmath.MathParagraph.Maths.Add();
//Adds a right subsuperscript equation
IOfficeMathRightScript officeMathRightScript = officeMath.Functions.Add(0,
MathFunctionType.RightSubSuperscript) as IOfficeMathRightScript;
//Sets false to align subscript and superscript horizontally
officeMathRightScript.IsSkipAlign = true;
//Adds run element for right subscript
IOfficeMathRunElement officeMathRunElement =
officeMathRightScript.Subscript.Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for right subscript
(officeMathRunElement.Item as WTextRange).Text = "1";
//Adds run element for right superscript
officeMathRunElement =
officeMathRightScript.Superscript.Functions.Add(0,
MathFunctionType.RunElement) as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for right superscript.
(officeMathRunElement.Item as WTextRange).Text = "n";
officeMathRunElement =
officeMathRightScript.Equation.Functions.Add(0, MathFunctionType.RunElement)
as IOfficeMathRunElement;
officeMathRunElement.Item = new WTextRange(document);
//Sets text for equation.
(officeMathRunElement.Item as WTextRange).Text = "Y";
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN
```

Modify existing equation

You can add or modify the text and formatting of existing mathematical equation in Word document. The following screenshots shows an existing mathematical equation in the input Word document.



The following code example shows how to modify an existing mathematical equation in the Word document.

C#

```
//Opens an existing Word document
WordDocument document = new WordDocument("Template.docx");
//Access the paragraph from Word document
WParagraph paragraph = document.LastSection.Body.ChildEntities[0] as
WParagraph;
//Access the mathematical equation from the paragraph
WMath math = paragraph.ChildEntities[2] as WMath;
//Access the radical equation
IOfficeMathRadical mathRadical = math.MathParagraph.Maths[0].Functions[1] as
IOfficeMathRadical;
//Access the fraction equation in radical
IOfficeMathFraction mathFraction = mathRadical.Equation.Functions[0] as
IOfficeMathFraction;
//Access the n-array equation in fraction
IOfficeMathNArray mathNArray = mathFraction.Numerator.Functions[0] as
IOfficeMathNArray;
//Access the math script in n-array
IOfficeMathScript mathScript = mathNArray.Equation.Functions[0] as
IOfficeMathScript;
//Access the delimiter in math script
IOfficeMathDelimiter mathDelimiter = mathScript.Equation.Functions[0] as
IOfficeMathDelimiter;
//Removes the delimiter
mathScript.Equation.Functions.Remove(mathDelimiter);
//Modifies the run element in math script
IOfficeMathRunElement MathParagraphItem =
mathScript.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
MathParagraphItem.Item = new WTextRange(document);
//Sets the text value
(MathParagraphItem.Item as WTextRange).Text = "x";
```

```
//Applies character format to the text
(MathParagraphItem.Item as WTextRange).CharacterFormat.Italic = true;
(MathParagraphItem.Item as WTextRange).CharacterFormat.FontSize = 20;
//Applies math format to the text
MathParagraphItem.MathFormat.Style = MathStyleType.Italic;
//Saves the word document
document.Save("Sample.docx");
//Close the word document
document.Close();
```

VB.NET

```
'Opens an existing Word document
Dim document As WordDocument = New WordDocument("Template.docx")
'Access the paragraph from Word document
Dim paragraph As WParagraph =
CType(document.LastSection.Body.ChildEntities(0), WParagraph)
'Access the mathematical equation from the paragraph
Dim math As WMath = CType(paragraph.ChildEntities(2), WMath)
'Access the radical equation
Dim mathRadical As IOOfficeMathRadical =
CType(math.MathParagraph.Maths(0).Functions(1), IOOfficeMathRadical)
'Access the fraction equation in radical
Dim mathFraction As IOOfficeMathFraction =
CType(mathRadical.Equation.Functions(0), IOOfficeMathFraction)
'Access the n-array equation in fraction
Dim mathNary As IOOfficeMathNArray =
CType(mathFraction.Numerator.Functions(0), IOOfficeMathNArray)
'Access the math script in n-array
Dim mathScript As IOOfficeMathScript = CType(mathNary.Equation.Functions(0),
IOOfficeMathScript)
'Access the delimiter in math script
Dim mathDelimiter As IOOfficeMathDelimiter =
CType(mathScript.Equation.Functions(0), IOOfficeMathDelimiter)
mathScript.Equation.Functions.Remove(mathDelimiter)
Dim MathParagraphItem As IOOfficeMathRunElement =
CType(mathScript.Equation.Functions.Add(MathFunctionType.RunElement),
IOOfficeMathRunElement)
MathParagraphItem.Item = New WTextRange(document)
'Modifies the math text value
CType(MathParagraphItem.Item, WTextRange).Text = "x"
'Applies character format to the text
CType(MathParagraphItem.Item, WTextRange).CharacterFormat.Italic = True
CType(MathParagraphItem.Item, WTextRange).CharacterFormat.FontSize = 20
'Applies math format to the text
MathParagraphItem.MathFormat.Style = MathStyleType.Italic
'Saves the word document
document.Save("Sample.docx")
'Close the word document
document.Close()
```

UWP

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing Word document
```

```

Stream fileStream =
assembly.GetManifestResourceStream("Sample.Assets.Template.docx");
//Access the paragraph from Word document
WParagraph paragraph = document.LastSection.Body.ChildEntities[0] as
WParagraph;
//Access the mathematical equation from the paragraph
WMath math = paragraph.ChildEntities[2] as WMath;
//Access the radical equation
IOfficeMathRadical mathRadical = math.MathParagraph.Maths[0].Functions[1] as
IOfficeMathRadical;
//Access the fraction equation in radical
IOfficeMathFraction mathFraction = mathRadical.Equation.Functions[0] as
IOfficeMathFraction;
//Access the n-array equation in fraction
IOfficeMathNArray mathNArray = mathFraction.Numerator.Functions[0] as
IOfficeMathNArray;
//Access the math script in n-array
IOfficeMathScript mathScript = mathNArray.Equation.Functions[0] as
IOfficeMathScript;
//Access the delimiter in math script
IOfficeMathDelimiter mathDelimiter = mathScript.Equation.Functions[0] as
IOfficeMathDelimiter;
//Removes the delimiter
mathScript.Equation.Functions.Remove(mathDelimiter);
//Modifies the run element in math script
IOfficeMathRunElement MathParagraphItem =
mathScript.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
MathParagraphItem.Item = new WTextRange(document);
//Sets the text value
(MathParagraphItem.Item as WTextRange).Text = "x";
//Applies character format to the text
(MathParagraphItem.Item as WTextRange).CharacterFormat.Italic = true;
(MathParagraphItem.Item as WTextRange).CharacterFormat.FontSize = 20;
//Applies math format to the text
MathParagraphItem.MathFormat.Style = MathStyleType.Italic;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word document file in local machine
Save(stream, "Sample.docx");
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Opens an existing Word document
FileStream fileStream = new FileStream("Template.docx", FileMode.Open,
FileAccess.ReadWrite);
//Loads the template document
WordDocument document = new WordDocument(fileStream, FormatTypeAutomatic);
//Access the paragraph from Word document
WParagraph paragraph = document.LastSection.Body.ChildEntities[0] as
WParagraph;

```

```

//Access the mathematical equation from the paragraph
WMath math = paragraph.ChildEntities[2] as WMath;
//Access the radical equation
IOfficeMathRadical mathRadical = math.MathParagraph.Maths[0].Functions[1] as
IOfficeMathRadical;
//Access the fraction equation in radical
IOfficeMathFraction mathFraction = mathRadical.Equation.Functions[0] as
IOfficeMathFraction;
//Access the n-array equation in fraction
IOfficeMathNArray mathNArray = mathFraction.Numerator.Functions[0] as
IOfficeMathNArray;
//Access the math script in n-array
IOfficeMathScript mathScript = mathNArray.Equation.Functions[0] as
IOfficeMathScript;
//Access the delimiter in math script
IOfficeMathDelimiter mathDelimiter = mathScript.Equation.Functions[0] as
IOfficeMathDelimiter;
//Removes the delimiter
mathScript.Equation.Functions.Remove(mathDelimiter);
//Modifies the run element in math script
IOfficeMathRunElement MathParagraphItem =
mathScript.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
MathParagraphItem.Item = new WTextRange(document);
//Sets the text value
(MathParagraphItem.Item as WTextRange).Text = "x";
//Applies character format to the text
(MathParagraphItem.Item as WTextRange).CharacterFormat.Italic = true;
(MathParagraphItem.Item as WTextRange).CharacterFormat.FontSize = 20;
//Applies math format to the text
MathParagraphItem.MathFormat.Style = MathStyleType.Italic;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");

```

XAMARIN

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Opens an existing Word document
Stream fileStream =
assembly.GetManifestResourceStream("Sample.Assets.Template.docx");
//Access the paragraph from Word document
WParagraph paragraph = document.LastSection.Body.ChildEntities[0] as
WParagraph;
//Access the mathematical equation from the paragraph
WMath math = paragraph.ChildEntities[2] as WMath;
//Access the radical equation
IOfficeMathRadical mathRadical = math.MathParagraph.Maths[0].Functions[1] as
IOfficeMathRadical;
//Access the fraction equation in radical

```

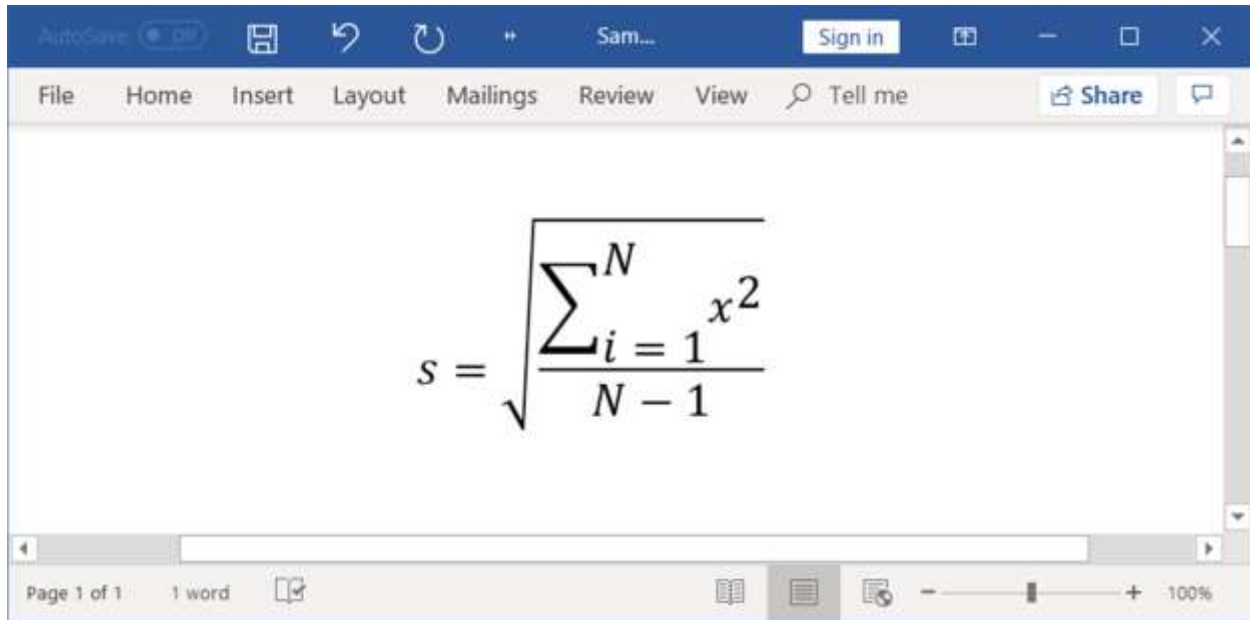


```

IOfficeMathFraction mathFraction = mathRadical.Equation.Functions[0] as
IOfficeMathFraction;
//Access the n-array equation in fraction
IOfficeMathNArray mathNArray = mathFraction.Numerator.Functions[0] as
IOfficeMathNArray;
//Access the math script in n-array
IOfficeMathScript mathScript = mathNArray.Equation.Functions[0] as
IOfficeMathScript;
//Access the delimiter in math script
IOfficeMathDelimiter mathDelimiter = mathScript.Equation.Functions[0] as
IOfficeMathDelimiter;
//Removes the delimiter
mathScript.Equation.Functions.Remove(mathDelimiter);
//Modifies the run element in math script
IOfficeMathRunElement MathParagraphItem =
mathScript.Equation.Functions.Add(MathFunctionType.RunElement) as
IOfficeMathRunElement;
MathParagraphItem.Item = new WTextRange(document);
//Sets the text value
(MathParagraphItem.Item as WTextRange).Text = "x";
//Applies character format to the text
(MathParagraphItem.Item as WTextRange).CharacterFormat.Italic = true;
(MathParagraphItem.Item as WTextRange).CharacterFormat.FontSize = 20;
//Applies math format to the text
MathParagraphItem.MathFormat.Style = MathStyleType.Italic;
//Saves and closes the Word document instance
MemoryStream stream = new MemoryStream();
//Saves the Word file to MemoryStream
document.Save(stream, FormatType.Docx);
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.doc
x", "application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
XAMARIN#helper-files-for-XAMARIN

```

By executing the above code example, it generates output Word document as follows.



Accepting or Rejecting Track Changes

It is used to keep track of the changes made to a Word document. It helps to maintain the record of author, name and time for every insertion, deletion, or modification in a document. This can be enabled by using the TrackChanges property of the Word document.

Note:

With this support, the changes made in the Word document by DocIO library cannot be tracked.

The following code example illustrates how to enable track changes of the document.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends text to the paragraph
IWTextRange text = paragraph.AppendText("This sample illustrates how to
track the changes made to the word document. ");
//Sets font name and size for text
text.CharacterFormat.FontName = "Times New Roman";
text.CharacterFormat.FontSize = 14;
text = paragraph.AppendText("This track changes is useful in shared
environment.");
text.CharacterFormat.FontSize = 12;
//Turns on the track changes option
document.TrackChanges = true;
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Appends text to the paragraph
Dim text As IWTextRange = paragraph.AppendText("This sample illustrates how
to track the changes made to the word document. ")
'Sets font name and size for text
text.CharacterFormat.FontName = "Times New Roman"
text.CharacterFormat.FontSize = 14
text = paragraph.AppendText("This track changes is useful in shared
environment.")
text.CharacterFormat.FontSize = 12
'Turns on the track changes option
document.TrackChanges = True
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

UWP

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends text to the paragraph
IWTextRange text = paragraph.AppendText("This sample illustrates how to
track the changes made to the word document. ");
//Sets font name and size for text
text.CharacterFormat.FontName = "Times New Roman";
text.CharacterFormat.FontSize = 14;
text = paragraph.AppendText("This track changes is useful in shared
environment.");
text.CharacterFormat.FontSize = 12;
//Turns on the track changes option
document.TrackChanges = true;
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "Sample.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();

```

```
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends text to the paragraph
IWTextRange text = paragraph.AppendText("This sample illustrates how to
track the changes made to the word document. ");
//Sets font name and size for text
text.CharacterFormat.FontName = "Times New Roman";
text.CharacterFormat.FontSize = 14;
text = paragraph.AppendText("This track changes is useful in shared
environment.");
text.CharacterFormat.FontSize = 12;
//Turns on the track changes option
document.TrackChanges = true;
//Saves the Word document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "Sample.docx");
```

XAMARIN

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Appends text to the paragraph
IWTextRange text = paragraph.AppendText("This sample illustrates how to
track the changes made to the word document. ");
//Sets font name and size for text
text.CharacterFormat.FontName = "Times New Roman";
text.CharacterFormat.FontSize = 14;
text = paragraph.AppendText("This track changes is useful in shared
environment.");
text.CharacterFormat.FontSize = 12;
//Turns on the track changes option
document.TrackChanges = true;
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.docx",
"application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

The changes made to the document can be accepted or rejected. The following code example illustrates how to accept or reject the changes made to the document.

C#

```
//Loads the template document
WordDocument document = new WordDocument("TrackChanges.docx",
FormatType.Docx);
//Accepts track changes of the document
if (document.HasChanges)
document.AcceptChanges();
//Saves and closes the document
document.Save("TrackChanges_Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Loads the template document
Dim document As New WordDocument("TrackChanges.docx", FormatType.Docx)
'Accepts track changes of the document
If document.HasChanges Then
document.AcceptChanges()
End If
'Saves and closes the document
document.Save("TrackChanges_Sample.docx", FormatType.Docx)
document.Close()
```

UWP

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Loads the template document
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Assets.TrackChanges.
docx"), FormatType.Docx);
//Accepts track changes of the document
if (document.HasChanges)
document.AcceptChanges();
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream, FormatType.Docx);
//Saves the stream as Word file in local machine
Save(stream, "TrackChanges_Sample.docx");
document.Close();
//Please refer the below link to save Word document in UWP platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
uwp#save-word-document-in-uwp
```

ASP.NET CORE

```
FileStream fileStreamPath = new FileStream(@"Data/TrackChanges.docx",
FileMode.Open, FileAccess.Read, FileShare.ReadWrite);
//Loads the template document
WordDocument document = new WordDocument(fileStreamPath, FormatType.Docx);
//Accepts track changes of the document
if (document.HasChanges)
```

```
document.AcceptChanges();
//Saves the Word file to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
stream.Position = 0;
//Download Word document in the browser
return File(stream, "application/msword", "TrackChanges_Sample.docx");
```

XAMARIN

```
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Loads the template document
WordDocument document = new
WordDocument(assembly.GetManifestResourceStream("Sample.Data.TrackChanges.do
cx"), FormatType.Docx);
//Accepts track changes of the document
if (document.HasChanges)
document.AcceptChanges();
MemoryStream stream = new MemoryStream();
//Saves the Word document to MemoryStream
document.Save(stream, FormatType.Docx);
//Closes the document
document.Close();
//Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TrackChanges_Sampl
e.docx", "application/msword", stream);
//Please download the helper files from the below link to save the stream as
file and open the file for viewing in Xamarin platform
//https://help.syncfusion.com/file-formats/docio/create-word-document-in-
xamarin#helper-files-for-xamarin
```

Conversion

Working with Document Conversions

The Essential DocIO converts documents from one format to another format. Each file format document can be categorized as flow layout document or fixed layout document.

Flow layout document

- A flow document is designed to "reflow content" depending on the application.
- Does not contain any information about the position of its content.
- Dynamically renders the content by application at run time.
- Example: DOC, DOCX, HTML, EPUB, RTF, and TEXT file formats.

Fixed layout document

- This format of fixed document is like "what you see is what you get".
- Maintains the fixed position for each content.
- Statically preserves the content in specified position.
- Example: Image and PDF.

Essential DocIO can convert various flow document as fixed document by using our layout engine. Following conversions are supported by Essential DocIO.

- Microsoft Word file format Conversions
- Word document to PDF
- Word document to Image
- RTF Conversions
- Text Conversions
- HTML Conversions
- Word document to ODT
- Word document to EPUB

Converting Word document to PDF

Essential DocIO allows you to convert a Word document into PDF with a few lines of code. For further information kindly refer [here](#).

Customizing the Word document to PDF conversion

Essential DocIO allows you to customize the Word to PDF conversion with the following options:

- Allows to embed the TrueType fonts used in the converted PDF
- Allows to determine the quality of the charts in the converted PDF
- Allows to determine the quality of the JPEG images in the converted PDF
- Allows to reduce the Main Memory usage in Word to PDF conversion by reusing the identical images.

For further information kindly refer [here](#).

Unsupported elements in Word to PDF conversion

Kindly refer the [here](#) for list of Unsupported elements in Word to PDF conversion

Rendering / Converting Word document to Image

Essential DocIO supports to convert the Word document to images using `RenderAsImages` method. For further information kindly refer [here](#).

RTF conversion

Essential DocIO supports to convert the RTF document into Word document and vice versa. For further information kindly refer [here](#).

HTML conversion

Essential DocIO supports converting the HTML file into Word document and vice versa. It supports only the HTML files that meet the validation either against XHTML 1.0 strict or XHTML 1.0 Transitional schema.

For further information kindly refer [here](#).

Customizing the HTML to Word conversion

You can customize the HTML to Word conversion with the following options:

- Validate the HTML string against XHTML 1.0 Strict and Transitional schema
- Insert the HTML string at the specified position of the document body contents
- Append HTML string to the specified paragraph

For further information kindly refer [here](#).

Customizing the Word to HTML conversion

You can customize the Word to HTML conversion with the following options:

- Extract the images used in the HTML document at the specified file directory
- Specify to export the header and footer of the Word document in the HTML
- Specify to consider Text Input field as a editable fields or text
- Specify the CSS style sheet type and its name

Note:

While exporting header and footer, DocIO exports the first section header content at the top of the HTML file and first section footer content at the end of the HTML file

For further information kindly refer [here](#).

Supported Document elements

Kindly refer the [here](#) for the document elements and attributes are supported by DocIO in Word to HTML and HTML to Word conversions.

Text file

Essential DocIO supports to convert the Word document into Text file and vice versa. For further information kindly refer [here](#).

Word to EPUB

Essential DocIO supports to convert the Word document into EPUB v2.0. It only supports in Windows Forms, UWP, WPF, ASP.NET Web and MVC platforms. For further information kindly refer [here](#).

Supported and Unsupported Features

This section describes the support and unsupported elements in the DocIO.

Word document features

Element	DOC, DOT		DOCX, DOTX, DOTM, DOCM, WordML		RTF	
	Read	Write	Read	Write	Read	Write
Built-in document properties	Yes	Yes	Yes	Yes	Yes	No
Custom document properties	Yes	Yes	Yes	Yes	No	No
Mail merge	Yes	Yes	Yes	Yes	Yes	Yes

Element	DOC, DOT		DOCX, DOTX, DOTM, DOCM, WordML		RTF	
Encryption and Decryption	Yes	Yes	Yesâ€™limited [except Word 2013]	Yesâ€™limited [except Word 2013]	N/A	N/A
View setup	Yes	Yes	Yes	Yes	Yes	No
Watermark	Yes	Yes	Yes	Yes	No	Yes
Font substitution table	Yes	Yes	Yes	Yes	Yes	No
Attached Template	No	No	Yes	Yes	No	No
Track changes	Yes	Yes â€™ limited [can only accept/reject]	Yes	Yes â€™ limited [can only accept/reject]	No	No
Themes	No	No	Yes	Yes	No	No
Document variables	Yes	Yes	Yes	Yes	No	No
Macros	No	No	Yes	Yes	No	No
Embedded fonts	No	No	Yes	Yes	No	No

Section features

Element	DOC, DOT		DOCX, DOTX, DOTM, DOCM, WordML		RTF	
	Read	Write	Read	Write	Read	Write
Headers & footers	Yes	Yes	Yes	Yes	Yes	Yes
Section break	Yes	Yes	Yes	Yes	Yes	Yes

Element	DOC, DOT		DOCX, DOTX, DOTM, DOCM, WordML		RTF	
Text columns	Yes	Yes	Yes	Yes	Yes	Yes
Page margin	Yes	Yes	Yes	Yes	Yes	Yes
Page border	Yes	Yes	Yes	Yes	No	Yes
Page orientation	Yes	Yes	Yes	Yes	Yes	Yes
Page size	Yes	Yes	Yes	Yes	Yes	Yes
Header & footer distance	Yes	Yes	Yes	Yes	No	No
Vertical alignment	Yes	Yes	Yes	Yes	Yes	Yes

Paragraph elements features

Element	DOC, DOT		DOCX, DOTX, DOTM, DOCM, WordML		RTF	
	Read	Write	Read	Write	Read	Write
Break “ Page break, column break and Line break	Yes	Yes	Yes	Yes	Yes	Yes
Symbols	Yes	Yes	Yes	Yes	No	Yes
Comments	Yes	Yes	Yes	Yes	Yes	Yes
Footnote and endnote	Yes	Yes	Yes	Yes	No	Yes
Pictures (.bmp, .jpg, .png, .emf, .tif and .gif)	Yes	Yes	Yes	Yes	Yes	Yes
Text boxes	Yes	Yes	Yes	Yes	No	No
OLE objects	Yes	Yes	Yes	Yes	No	Yes

Element	DOC, DOT		DOCX, DOTX, DOTM, DOCM, WordML		RTF	
Predefined shapes	No	No	Yes	Yes	No	No
Custom shapes	No	No	No	No	No	No
Grouped shapes	No	No	Yes	Yes	No	No
Equation	No	No	No	No	No	No
SmartArt	No	No	No	No	No	No
WordArt	No	No	No	No	No	No
Bookmark	Yes	Yes	Yes	Yes	Yes	Yes
Hyperlink	Yes	Yes	Yes	Yes	Yes	Yes
Chart	No	No	Yes	Yes	No	No
Fields	Yes	Yes	Yes	Yes	Yes	Yes
Form Fields- TextInput, CheckBox & DropDown	Yes	Yes	Yes	Yes	Yes	Yes

Paragraph features

Element	DOC, DOT		DOCX, DOTX, DOTM, DOCM		RTF	
	Read	Write	Read	Write	Read	Write
Paragraph style	Yes	Yes	Yes	Yes	Yes	Yes
Alignment “ left, right, center and justify	Yes	Yes	Yes	Yes	Yes	Yes
Right to left paragraph	Yes	Yes	Yes	Yes	Yes	Yes

Element	DOC, DOT		DOCX, DOTX, DOTM, DOCM		RTF	
Lists - bullets and numbers	Yes	Yes	Yes	Yes	Yes	Yes
Run properties for the paragraph mark	Yes	Yes	Yes	Yes	Yes	Yes
Suppress hyphenation	No	No	No	No	No	No
Indents “ left, right, first line & hanging	Yes	Yes	Yes	Yes	Yes	Yes
Spacing “ before, after & auto	Yes	Yes	Yes	Yes	Yes	Yes
Keeps and breaks	Yes	Yes	Yes	Yes	Yes	Yes
Text frames	Yes	Yes	Yes	Yes	Yes	Yes
Tabs	Yes	Yes	Yes	Yes	Yes	Yes
Borders	Yes	Yes	Yes	Yes	Yes	Yes
Shading	Yes	Yes	Yes	Yes	Yes	Yes

Style features

Element	DOC, DOT		DOCX, DOTX, DOTM, DOCM, WordML		RTF	
	Read	Write	Read	Write	Read	Write
Built-in paragraph style	Yes	Yes	Yes	Yes	Yes	Yes
Custom paragraph style	Yes	Yes	Yes	Yes	Yes	Yes

Element	DOC, DOT		DOCX, DOTX, DOTM, DOCM, WordML		RTF	
Built-in character style	Yes	Yes	Yes	Yes	Yes	Yes
Custom character style	Yes	Yes	Yes	Yes	Yes	Yes
Built-in list style	Yes	Yes	Yes	Yes	Yes	Yes
Custom list style	Yes	Yes	Yes	Yes	Yes	Yes
Built-in table style	No	No	Yes	Yes	No	No
Custom table style	No	No	Yes	Yes	No	No

Table features

Element	DOC, DOT		DOCX, DOTX, DOTM, DOCM, WordML		RTF	
	Read	Write	Read	Write	Read	Write
Nested tables	Yes	Yes	Yes	Yes	Yes	Yes
Right to left tables	Yes	Yes	Yes	Yes	Yes	Yes
Table style	No	No	Yes	Yes	No	No
Conditional formatting style	No	No	Yes	Yes	No	No
Table alignment	Yes	Yes	Yes	Yes	Yes	Yes
Table indent	Yes	Yes	Yes	Yes	Yes	Yes
Allow AutoFit	Yes	Yes	Yes	Yes	Yes	Yes
Cell margins	Yes	Yes	Yes	Yes	Yes	Yes
Cell spacing	Yes	Yes	Yes	Yes	Yes	Yes

Element	DOC, DOT		DOCX, DOTX, DOTM, DOCM, WordML		RTF	
Preferred table width	Yes	Yes	Yes	Yes	Yes	Yes
Table shading	Yes	Yes	Yes	Yes	Yes	Yes
Table borders	Yes	Yes	Yes	Yes	Yes	Yes
Floating tables	Yes	Yes	Yes	Yes	Yes	Yes
Allow row to break across pages	Yes	Yes	Yes	Yes	Yes	Yes
Repeat as header row	Yes	Yes	Yes	Yes	Yes	Yes
Height	Yes	Yes	Yes	Yes	Yes	Yes
Height type	Yes	Yes	Yes	Yes	Yes	Yes
Cell borders	Yes	Yes	Yes	Yes	Yes	Yes
Cell shading	Yes	Yes	Yes	Yes	Yes	Yes
Wrap text	Yes	Yes	Yes	Yes	Yes	Yes
Fit text	Yes	Yes	Yes	Yes	Yes	Yes
Horizontal & vertical merged cells	Yes	Yes	Yes	Yes	Yes	Yes
Vertical alignment	Yes	Yes	Yes	Yes	Yes	Yes
Text direction	Yes	Yes	Yes	Yes	Yes	Yes

Text features

Element	DOC, DOT	DOCX, DOTX, DOTM, DOCM, WordML	RTF
---------	----------	--------------------------------	-----

	Read	Write	Read	Write	Read	Write
Character style	Yes	Yes	Yes	Yes	Yes	Yes
Color	Yes	Yes	Yes	Yes	Yes	Yes
Highlight color	Yes	Yes	Yes	Yes	Yes	Yes
Language	Yes	Yes	Yes	Yes	Yes	Yes
Border	Yes	Yes	Yes	Yes	No	Yes
Shading	Yes	Yes	Yes	Yes	Yes	Yes
Font	Yes	Yes	Yes	Yes	Yes	Yes
Underline	Yes	Yes	Yes	Yes	Yes	Yes
Strikethrough	Yes	Yes	Yes	Yes	Yes	Yes
Double strikethrough	Yes	Yes	Yes	Yes	Yes	Yes
Subscript/Superscript	Yes	Yes	Yes	Yes	Yes	Yes
Small caps	Yes	Yes	Yes	Yes	Yes	Yes
All caps	Yes	Yes	Yes	Yes	Yes	Yes
Hidden text	Yes	Yes	Yes	Yes	Yes	Yes
Scale	Yes	Yes	Yes	Yes	Yes	Yes
Expanded / Compressed	Yes	Yes	Yes	Yes	Yes	No
Vertical Position	Yes	Yes	Yes	Yes	No	Yes

Blazor supported features

Feature	Server side and hosted application	Client side application
Pictures	Yes	Yes
Chart	Yes	Yes

AutoShapes	Yes	Yes
Group Shapes	Yes	Yes
Content control	Yes	Yes
Bookmarks	Yes	Yes
OLE Object	Yes	Yes
Form Fields	Yes	Yes
Watermark	Yes	Yes
Break	Yes	Yes
RTL	Yes	Yes
Header and Footer	Yes	Yes
Footnotes and Endnotes	Yes	Yes
Styles	Yes	Yes
Table Styles	Yes	Yes
Document Settings	Yes	Yes
Mail Merge	Yes	Yes
Updating Document Fields	Yes	Yes
Find and Replace	Yes	Yes
Bookmark Navigation	Yes	Yes
Clone and Merge	Yes	Yes
Formatting Table	Yes	Yes
Macro Preservation	Yes	Yes
Document Protection	Yes	Yes
Word to PDF	Yes	No

RTF to Word	Yes	Yes
Word to WordML	Yes	Yes
WordML to Word	Yes	Yes
Word to ODT	Yes	Yes

FAQ/How to

How to modify an existing style?

The following code illustrates how to modify the built-in style while creating new Word document.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Creates built-in style and modifies its properties
Style style = Style.CreateBuiltinStyle(BuiltinStyle.Heading1, document) as
Style;
style.CharacterFormat.Italic = true;
style.CharacterFormat.TextColor = Color.DarkGreen;
//Adds it to the styles collection
document.Styles.Add(style);
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
IWTextRange text = paragraph.AppendText("A built-in style is modified and is
applied to this paragraph.");
//Applies the new style to paragraph
paragraph.ApplyStyle(style.Name);
//Saves the Word document
document.Save("Sample.docx", FormatType.Docx);
//Closes the document
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Creates built-in style and modifies its properties
Dim style__1 As Style =
TryCast(Style.CreateBuiltinStyle(BuiltinStyle.Heading1, document), Style)
style__1.CharacterFormat.Italic = True
style__1.CharacterFormat.TextColor = Color.DarkGreen
'Adds it to the styles collection
document.Styles.Add(style__1)
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
```

```

Dim text As ITextRange = paragraph.AppendText("A built-in style is modified
and is applied to this paragraph.")
'Applies the new style to paragraph
paragraph.ApplyStyle(style__1.Name)
'Saves the Word document
document.Save("Sample.docx", FormatType.Docx)
'Closes the document
document.Close()

```

How to open a document from stream using DocIO?

A document can be opened as stream by using `HttpWebResponse`. This stream does not support seek operation and so the contents should be read manually to get the position and length of the stream. The following code illustrates how to load the document from stream.

C#

```

//Gets the document as stream
HttpRequest request =
(HttpRequest)WebRequest.Create("https://www.swiftview.com/tech/letterlega
15.doc");
HttpWebResponse response = (HttpWebResponse)request.GetResponse();
Stream stream = response.GetResponseStream();
//Converts it to byte array
byte[] buffer = ReadFully(stream, 32768);
//Stores bytes into the memory stream.
MemoryStream ms = new MemoryStream();
ms.Write(buffer, 0, buffer.Length);
ms.Seek(0, SeekOrigin.Begin);
stream.Close();
//Creates a new document.
WordDocument document = new WordDocument();
//Opens the template document from the MemoryStream.
document.Open(ms, FormatType.Doc);
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Gets the document as stream
Dim request As HttpRequest =
DirectCast(WebRequest.Create("https://www.swiftview.com/tech/letterlegal5.do
c"), HttpRequest)
Dim response As HttpWebResponse = DirectCast(request.GetResponse(),
HttpWebResponse)
Dim stream As Stream = response.GetResponseStream()
'Converts it to byte array
Dim buffer As Byte() = ReadFully(stream, 32768)
'Stores bytes into the memory stream.
Dim ms As New MemoryStream()
ms.Write(buffer, 0, buffer.Length)
ms.Seek(0, SeekOrigin.Begin)
stream.Close()
'Creates a new document.
Dim document As New WordDocument ()

```

```
'Opens the template document from the MemoryStream.
document.Open(ms, FormatType.Doc)
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

The following code illustrates the method used to read the stream and convert the stream to bytes.

C#

```
public static byte[] ReadFully(Stream stream, int initialLength)
{
    //When an unhelpful initial length has been passed, just use 32K.
    if (initialLength < 1)
    {
        initialLength = 32768;
    }
    byte[] buffer = new byte[initialLength];
    int read = 0;
    int chunk;
    while ((chunk = stream.Read(buffer, read, buffer.Length - read)) > 0)
    {
        read += chunk;
        //After reaching the end of the buffer, check and see whether you can find
        //any information.
        if (read == buffer.Length)
        {
            int nextByte = stream.ReadByte();
            //End of stream? Then, you are done.
            if (nextByte == -1)
            {
                return buffer;
            }
            //Resize the buffer, put in the byte you have just read, and continue.
            byte[] newBuffer = new byte[buffer.Length * 2];
            Array.Copy(buffer, newBuffer, buffer.Length);
            newBuffer[read] = (byte)nextByte;
            buffer = newBuffer;
            read++;
        }
    }
    //Buffer is now too big. Shrink it.
    byte[] ret = new byte[read];
    Array.Copy(buffer, ret, read);
    return ret;
}
```

VB.NET

```
Public Shared Function ReadFully(stream As Stream, initialLength As Integer)
    As Byte()
    'When an unhelpful initial length has been passed, just use 32K.
    If initialLength < 1 Then
        initialLength = 32768
    End If
    Dim buffer As Byte() = New Byte(initialLength - 1) {}
```

```

Dim read As Integer = 0
Dim chunk As Integer
chunk = stream.Read(buffer, read, buffer.Length - read)
While (chunk > 0)
    read += chunk
    'After reaching the end of the buffer, check and see whether you can find
    any information.
    If read = buffer.Length Then
        Dim nextByte As Integer = stream.ReadByte()
        'End of stream? Then, you are done.
        If nextByte = -1 Then
            Return buffer
        End If
        'Resize the buffer, put in the byte you have just read, and continue.
        Dim newBuffer As Byte() = New Byte(buffer.Length * 2 - 1) {}
        Array.Copy(buffer, newBuffer, buffer.Length)
        newBuffer(read) = CByte(nextByte)
        buffer = newBuffer
        read += 1
    End If
End While
'Buffer is now too big. Shrink it.
Dim ret As Byte() = New Byte(read - 1) {}
Array.Copy(buffer, ret, read)
Return ret
End Function

```

How to set OpenType Font Features?

The Open type features provide special effects for the text. This feature is specific to Word 2010 and later version documents. The OpenType features includes the following:

- Ligatures – combination of characters, written as glyph
- Use Contextual Alternates – combination of letters based on surrounding characters
- Number spacing – specifies number width
- Number forms – specifies number height
- Stylistic sets – specifies the look of the text, based on the font used

The following code illustrates how to set ligature types for text.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new text
IWTextRange text = paragraph.AppendText("Text to describe discretionary
ligatures");
//Sets ligature type
text.CharacterFormat.Ligatures = LigatureType.Discretionary;
text.CharacterFormat.FontName = "Arial";
paragraph = section.AddParagraph();

```

```

text = paragraph.AppendText("Text to describe contextual ligatures");
text.CharacterFormat.Ligatures = LigatureType.Contextual;
text.CharacterFormat.FontName = "Arial";
paragraph = section.AddParagraph();
text = paragraph.AppendText("Text to describe historical ligatures");
text.CharacterFormat.Ligatures = LigatureType.Historical;
text.CharacterFormat.FontName = "Arial";
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Adds new text
Dim text As IWTextRange = paragraph.AppendText("Text to describe
discretional ligatures")
'Sets ligature type as Discretional
text.CharacterFormat.Ligatures = LigatureType.Discretional
text.CharacterFormat.FontName = "Arial"
paragraph = section.AddParagraph()
text = paragraph.AppendText("Text to describe contextual ligatures")
'Sets ligature type as Contextual
text.CharacterFormat.Ligatures = LigatureType.Contextual
text.CharacterFormat.FontName = "Arial"
paragraph = section.AddParagraph()
text = paragraph.AppendText("Text to describe historical ligatures")
'Sets ligature type as Historical
text.CharacterFormat.Ligatures = LigatureType.Historical
text.CharacterFormat.FontName = "Arial"
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

The following code example illustrates how to set contextual alternates.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new text
IWTextRange text = paragraph.AppendText("Text to describe contextual
alternates");
text.CharacterFormat.FontName = "Segoe Script";
//Sets contextual alternates
text.CharacterFormat.UseContextualAlternates = true;
paragraph = section.AddParagraph();

```

```
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Adds new text
Dim text As IWTextRange = paragraph.AppendText("Text to describe contextual
alternates")
text.CharacterFormat.FontName = "Segoe Script"
'Sets contextual alternates
text.CharacterFormat.UseContextualAlternates = True
paragraph = section.AddParagraph()
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

The following code example illustrates how to set number spacing.

C#

```
//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new text
IWTextRange text = paragraph.AppendText("Numbers to describe tabular number
spacing 0123456789");
text.CharacterFormat.FontName = "Calibri";
//Sets number spacing
text.CharacterFormat.NumberSpacing = NumberSpacingType.Tabular;
paragraph = section.AddParagraph();
text = paragraph.AppendText("Numbers to describe proportional number spacing
0123456789");
text.CharacterFormat.FontName = "Calibri";
//Sets number spacing
text.CharacterFormat.NumberSpacing = NumberSpacingType.Proportional;
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
```

```

'Adds new paragraph to the section
Dim paragraph As IParagraph = section.AddParagraph()
'Adds new text
Dim text As ITextRange = paragraph.AppendText("Numbers to describe tabular
number spacing 0123456789")
text.CharacterFormat.FontName = "Calibri"
'Sets number spacing
text.CharacterFormat.NumberSpacing = NumberSpacingType.Tabular
paragraph = section.AddParagraph()
text = paragraph.AppendText("Numbers to describe proportional number spacing
0123456789")
text.CharacterFormat.FontName = "Calibri"
'Sets number spacing
text.CharacterFormat.NumberSpacing = NumberSpacingType.Proportional
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

The following code example illustrates how to set number style.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IParagraph paragraph = section.AddParagraph();
//Adds new text
ITextRange text = paragraph.AppendText("Numbers to describe oldstyle number
form 0123456789");
text.CharacterFormat.FontName = "Calibri";
//Sets number style
text.CharacterFormat.NumberForm = NumberFormType.OldStyle;
paragraph = section.AddParagraph();
text = paragraph.AppendText("Numbers to describe lining number form
0123456789");
text.CharacterFormat.FontName = "Calibri";
//Sets number style
text.CharacterFormat.NumberForm = NumberFormType.Lining;
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IParagraph = section.AddParagraph()
'Adds new text
Dim text As ITextRange = paragraph.AppendText("Numbers to describe oldstyle
number form 0123456789")
text.CharacterFormat.FontName = "Calibri"

```

```

'Sets number style
text.CharacterFormat.NumberForm = NumberFormType.OldStyle
paragraph = section.AddParagraph()
text = paragraph.AppendText("Numbers to describe lining number form
0123456789")
text.CharacterFormat.FontName = "Calibri"
'Sets number style
text.CharacterFormat.NumberForm = NumberFormType.Lining
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

The following code example illustrates how to set different styles for the text.

C#

```

//Creates a new Word document
WordDocument document = new WordDocument();
//Adds new section to the document
IWSection section = document.AddSection();
//Adds new paragraph to the section
IWParagraph paragraph = section.AddParagraph();
//Adds new text
IWTextRange text = paragraph.AppendText("Text to describe stylistic sets");
text.CharacterFormat.FontName = "Gabriola";
//Sets stylistic set
text.CharacterFormat.StylisticSet = StylisticSetType.StylisticSet06;
paragraph = section.AddParagraph();
//Adds new text
text = paragraph.AppendText("Text to describe stylistic sets");
text.CharacterFormat.FontName = "Gabriola";
//Sets stylistic set
text.CharacterFormat.StylisticSet = StylisticSetType.StylisticSet15;
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates a new Word document
Dim document As New WordDocument()
'Adds new section to the document
Dim section As IWSection = document.AddSection()
'Adds new paragraph to the section
Dim paragraph As IWParagraph = section.AddParagraph()
'Adds new text
Dim text As IWTextRange = paragraph.AppendText("Text to describe stylistic
sets")
text.CharacterFormat.FontName = "Gabriola"
'Sets stylistic set
text.CharacterFormat.StylisticSet = StylisticSetType.StylisticSet06
paragraph = section.AddParagraph()
'Adds new text
text = paragraph.AppendText("Text to describe stylistic sets")
text.CharacterFormat.FontName = "Gabriola"
'Sets stylistic set

```



```
text.CharacterFormat.StylisticSet = StylisticSetType.StylisticSet15
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

How to attach a Template to a Word document?

The following code illustrates how to set the template for the document.

C#

```
//Loads a source document
WordDocument document = new WordDocument("Template.docx");
//Attaches the template document to the source document
document.AttachedTemplate.Path = @"D:\Data\Template.docx";
//Updates the styles of the document from the attached template each time
the document is opened
document.UpdateStylesOnOpen = true;
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Loads a source document
Dim document As New WordDocument("Template.docx")
'Attaches the template document to the source document
document.AttachedTemplate.Path = "D:\Data\Template.docx"
'Updates the styles of the document from the attached template each time the
document is opened
document.UpdateStylesOnOpen = True
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

How to insert a DataTable in a Word document?

You can create new table in a Word document and copy the contents from data table. The following code illustrates how to insert a data table as table in a Word document.

C#

```
//Creates new Word document
WordDocument document = new WordDocument();
//Creates new data set and data table
DataSet dataset = new DataSet();
GetDataTable(dataset);
DataTable datatable = new DataTable();
datatable = dataset.Tables[0];
//Adds new section
IWSection section = document.AddSection();
//Adds new table
IWTable table = section.AddTable();
//Adds new row to the table
WTableRow row = table.AddRow();
foreach (DataColumn datacolumn in datatable.Columns)
{
```

```

//Sets the column names for the table from the data table column names and cell width
WTableCell cell = row.AddCell();
cell.AddParagraph().AppendText(datacolumn.ColumnName);
cell.Width = 150;
}
//Iterates through data table rows
foreach (DataRow datarow in datatable.Rows)
{
    //Adds new row to the table
    row = table.AddRow(true, false);
    foreach (object datacolumn in datarow.ItemArray)
    {
        //Adds new cell
        WTableCell cell = row.AddCell();
        //Adds contents from the data table to the table cell
        cell.AddParagraph().AppendText(datacolumn.ToString());
    }
}
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates new Word document
Dim document As New WordDocument()
'Creates new data set and data table
Dim dataset As New DataSet()
GetDataTable(dataset)
Dim datatable As New DataTable()
datatable = dataset.Tables(0)
'Adds new section
Dim section As IWSection = document.AddSection()
'Adds new table
Dim table As IWTable = section.AddTable()
'Adds new row to the table
Dim row As WTableRow = table.AddRow()
For Each datacolumn As DataColumn In datatable.Columns
    'Sets the column names for the table from the data table column names and cell width
    Dim cell As WTableCell = row.AddCell()
    cell.AddParagraph().AppendText(datacolumn.ColumnName)
    cell.Width = 150
Next
'Iterates through data table rows
For Each datarow As DataRow In datatable.Rows
    'Adds new row to the table
    row = table.AddRow(True, False)
    For Each datacolumn As Object In datarow.ItemArray
        'Adds new cell
        Dim cell As WTableCell = row.AddCell()
        'Adds contents from the data table to the table cell
        cell.AddParagraph().AppendText(datacolumn.ToString())
    Next
Next

```

```
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

The following code illustrates the method to get data table.

C#

```
private void GetDataTable(DataSet dataset)
{
    // List of syncfusion products.
    string[] products = { "DocIO", "PDF", "XlsIO" };
    // Adds new Tables to the data set.
    DataRow row;
    dataset.Tables.Add();
    // Adds fields to the Products table.
    dataset.Tables[0].TableName = "Products";
    dataset.Tables[0].Columns.Add("ProductName");
    dataset.Tables[0].Columns.Add("Binary");
    dataset.Tables[0].Columns.Add("Source");
    // Inserts values to the tables.
    foreach (string product in products)
    {
        row = dataset.Tables["Products"].NewRow();
        row["ProductName"] = string.Concat("Essential ", product);
        row["Binary"] = "$895.00";
        row["Source"] = "$1,295.00";
        dataset.Tables["Products"].Rows.Add(row);
    }
}
```

VB.NET

```
Private Sub GetDataTable(dataset As DataSet)
    'List of syncfusion products.
    Dim products As String() = {"DocIO", "PDF", "XlsIO"}
    'Adds new Tables to the data set.
    Dim row As DataRow
    dataset.Tables.Add()
    'Adds fields to the Products table.
    dataset.Tables(0).TableName = "Products"
    dataset.Tables(0).Columns.Add("ProductName")
    dataset.Tables(0).Columns.Add("Binary")
    dataset.Tables(0).Columns.Add("Source")
    'Inserts values to the tables.
    For Each product As String In products
        row = dataset.Tables("Products").NewRow()
        row("ProductName") = String.Concat("Essential ", product)
        row("Binary") = "$895.00"
        row("Source") = "$1,295.00"
        dataset.Tables("Products").Rows.Add(row)
    Next
End Sub
```

How to insert a table from HTML string in Word document?

An HTML string can be inserted to the Word document at text body or paragraph. The following code illustrates how to insert a table to the document from the HTML string.

C#

```
//Loads the template document
WordDocument document = new WordDocument("Template.docx");
//Gets the text body
WTextBody textbody = document.Sections[0].Body;
//Html string that represents table with two rows and two columns
string htmlString = " <table border='1'><tr><td><p>First Row First Cell</p></td><td><p>First Row Second Cell</p></td></tr><tr><td><p>Second Row First Cell</p></td><td><p>Second Row Second Cell</p></td></tr></table> ";
//Inserts the string to the text body
textbody.InsertXHTML(htmlString);
//Saves and closes the document
document.Save("Sample.docx");
document.Close();
```

VB.NET

```
'Loads the template document
Dim document As New WordDocument("Template.docx")
'Gets the text body
Dim textbody As WTextBody = document.Sections(0).Body
'Html string that represents table with two rows and two columns
Dim htmlString As String = " <table border='1'><tr><td><p>First Row First Cell</p></td><td><p>First Row Second Cell</p></td></tr><tr><td><p>Second Row First Cell</p></td><td><p>Second Row Second Cell</p></td></tr></table> "
'Inserts the string to the text body
textbody.InsertXHTML(htmlString)
'Saves and closes the document
document.Save("Sample.docx")
document.Close()
```

How to set table cell width?

Each cell in the table can have its own width. The following code illustrates how to set the width of the cell.

C#

```
//Creates new word document
WordDocument document = new WordDocument("Template.docx");
//Gets the text body of first section
WTextBody textbody = document.Sections[0].Body;
//Gets the table
IWTable table = textbody.Tables[0];
//Iterates through table rows
foreach (WTableRow row in table.Rows)
{
    //Sets width for cells
    for (int i = 0; i < row.Cells.Count; i++)
    {
        WTableCell cell = row.Cells[i];
```

```

if (i % 2 == 0)
    //Sets width as 100 for cells in even column
    cell.Width = 100;
else
    //Sets width as 150 for cell in odd column
    cell.Width = 150;
}
}
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Creates new word document
Dim document As New WordDocument("Template.docx")
'Gets the text body of first section
Dim textbody As WTextBody = document.Sections(0).Body
'Gets the table
Dim table As ITable = textbody.Tables(0)
'Iterates through table rows
For Each row As WTableRow In table.Rows
    'Sets width for cells
    For i As Integer = 0 To row.Cells.Count - 1
        Dim cell As WTableCell = row.Cells(i)
        If i Mod 2 = 0 Then
            'Sets width as 100 for cells in even column
            cell.Width = 100
        Else
            'Sets width as 150 for cell in odd column
            cell.Width = 150
        End If
    Next
Next
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

How to position a table in a Word document?

You can position a table in a Word document by setting position properties. The following code illustrates how to set position properties for a table.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Template.docx");
//Gets the text body of first section
WTextBody textbody = document.Sections[0].Body;
//Gets the table
ITable table = textbody.Tables[0];
//Sets the horizontal and vertical position for table
table.TableFormat.Positioning.HorizPosition = 40;
table.TableFormat.Positioning.VertPosition = 100;
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);

```

```
document.Close();
```

VB.NET

```
'Loads the template document
Dim document As New WordDocument("Template.docx")
'Gets the text body of first section
Dim textbody As WTextBody = document.Sections(0).Body
'Gets the table
Dim table As ITable = textbody.Tables(0)
'Sets the horizontal and vertical position for table
table.TableFormat.Positioning.HorizPosition = 40
table.TableFormat.Positioning.VertPosition = 100
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

How to set the text direction to a table in Word document?

The contents of the table cell can be in vertical or horizontal direction. Each cell content can have different text direction. The following code illustrates how to set the text direction for the text in the table.

C#

```
//Loads the template document
WordDocument document = new WordDocument("Template.docx");
//Gets the text body of first section
WTextBody textbody = document.Sections[0].Body;
//Gets the table
ITable table = textbody.Tables[0];
//Iterates through table rows
foreach (WTableRow row in table.Rows)
{
    foreach (WTableCell cell in row.Cells)
    {
        //Sets the text direction for the contents
        cell.CellFormat.TextDirection = Syncfusion.DocIO.DLS.TextDirection.Vertical;
    }
}
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Loads the template document
Dim document As New WordDocument("Template.docx")
'Gets the text body of first section
Dim textbody As WTextBody = document.Sections(0).Body
'Gets the table
Dim table As ITable = textbody.Tables(0)
'Iterates through table rows
For Each row As WTableRow In table.Rows
    For Each cell As WTableCell In row.Cells
        'Sets the text direction for the contents
```

```

cell.CellFormat.TextDirection = Syncfusion.DocIO.DLS.TextDirection.Vertical
Next
Next
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

How to extract the images in the document?

The following code illustrates how to extract the images in the document.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Template.docx");
WTextBody textbody = document.Sections[0].Body;
Image image;
int i = 1;
//Iterates through the paragraphs
foreach (WParagraph paragraph in textbody.Paragraphs)
{
    //Iterates through the paragraph items
    foreach (ParagraphItem item in paragraph.ChildEntities)
    {
        //Gets the picture and saves it into specified location
        switch (item.EntityType)
        {
            case EntityType.Picture:
                WPicture picture = item as WPicture;
                image = picture.Image;
                image.Save(@"D:\Data\Image" + i + ".jpeg", ImageFormat.Jpeg);
                i++;
                break;
            default:
                break;
        }
    }
}
//Closes the document
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx")
Dim textbody As WTextBody = document.Sections(0).Body
Dim image As Image
Dim i As Integer = 1
'Iterates through the paragraphs
For Each paragraph As WParagraph In textbody.Paragraphs
    'Iterates through the paragraph items
    For Each item As ParagraphItem In paragraph.ChildEntities
        'Gets the picture and saves it into specified location
        Select Case item.EntityType
            Case EntityType.Picture
                Dim picture As WPicture = TryCast(item, WPicture)
                image = picture.Image

```

```

image.Save("D:\Data\Image" & i & ".jpeg", ImageFormat.Jpeg)
i += 1
Exit Select
Case Else
Exit Select
End Select
Next
Next
'Close the document
document.Close()

```

The images in the document can be extracted into a specific location when exporting it to HTML file. The following code illustrates how to extract images.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Template.docx");
//Sets the location to extract images
document.SaveOptions.HtmlExportImagesFolder = @"D:\Data\";
//Saves the document as html file
HTMLExport export = new HTMLExport();
export.SaveAsXhtml(document, "Template.html");
//Closes the document
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx")
'Sets the location to extract images
document.SaveOptions.HtmlExportImagesFolder = "D:\Data\"
'Saves the document as html file
Dim export As New HTMLExport()
export.SaveAsXhtml(document, "Template.html")
'Closes the document
document.Close()

```

How to remove headers and footers from the document?

The following code illustrates how to remove the header contents from the document.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Template.docx", FormatType.Docx);
//Iterates through the sections
foreach (WSection section in document.Sections)
{
HeaderFooter header;
//Gets even footer of current section
header = section.HeadersFooters[HeaderFooterType.EvenHeader];
//Removes even footer
header.ChildEntities.Clear();
//Gets odd footer of current section
header = section.HeadersFooters[HeaderFooterType.OddHeader];

```



```

//Removes odd footer
header.ChildEntities.Clear();
//Gets first page footer
header = section.HeadersFooters[HeaderFooterType.FirstPageHeader];
//Removes first page footer
header.ChildEntities.Clear();
}
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx", FormatType.Docx)
'Iterates through the sections
For Each section As WSection In document.Sections
Dim header As HeaderFooter
'Gets even footer of current section
header = section.HeadersFooters(HeaderFooterType.EvenHeader)
'Removes even footer
header.ChildEntities.Clear()
'Gets odd footer of current section
header = section.HeadersFooters(HeaderFooterType.OddHeader)
'Removes odd footer
header.ChildEntities.Clear()
'Gets first page footer
header = section.HeadersFooters(HeaderFooterType.FirstPageHeader)
'Removes first page footer
header.ChildEntities.Clear()
Next
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

The following code illustrates how to remove the footer contents from the document.

C#

```

//Loads the template document
WordDocument document = new WordDocument("Template.docx");
//Iterates through the sections
foreach (WSection section in document.Sections)
{
HeaderFooter footer;
//Gets even footer of current section
footer = section.HeadersFooters[HeaderFooterType.EvenFooter];
//Removes even footer
footer.ChildEntities.Clear();
//Gets odd footer of current section
footer = section.HeadersFooters[HeaderFooterType.OddFooter];
//Removes odd footer
footer.ChildEntities.Clear();
//Gets first page footer
footer = section.HeadersFooters[HeaderFooterType.FirstPageFooter];
//Removes first page footer
}

```

```

footer.ChildEntities.Clear();
}
//Saves and closes the document
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```

'Loads the template document
Dim document As New WordDocument("Template.docx")
'Iterates through the sections
For Each section As WSection In document.Sections
Dim footer As HeaderFooter
'Gets even footer of current section
footer = section.HeadersFooters(HeaderFooterType.EvenFooter)
'Removes even footer
footer.ChildEntities.Clear()
'Gets odd footer of current section
footer = section.HeadersFooters(HeaderFooterType.OddFooter)
'Removes odd footer
footer.ChildEntities.Clear()
'Gets first page footer
footer = section.HeadersFooters(HeaderFooterType.FirstPageFooter)
'Removes first page footer
footer.ChildEntities.Clear()
Next
'Saves and closes the document
document.Save("Sample.docx", FormatType.Docx)
document.Close()

```

Which units does Essential DocIO uses for measurement properties such as size, margins, etc, in a Word document?

Essential DocIO library uses Points for measurement properties in a Word document.

Could not find Syncfusion.OfficeChartToImageConverter assembly in .NET 3.5 Framework, does it mean there is no support for chart conversion in this Framework?

Yes, OfficeChartToImageConverter assembly is not supported in .NET 3.5 Framework and it is available in .NET 4.0 Framework.

Can the chart data be refreshed?

Yes, Essential DocIO supports refreshing the chart data. For more details, refer [Working with charts](#)

Is it possible to convert 3D charts to PDF or image?

Current version of the DocIO library does not provide support for converting 3D charts to PDF or image format.

Is it possible to specify PDF conformance level in Word to PDF conversion?

Yes, you can specify the PDF conformance level in Word to PDF conversion. For more details, refer [PDF Conformance](#)

Migration from Microsoft Office Automation to Essential DocIO

Mail merge

The Mail merge feature can be used to generate reports and letters in Microsoft Word. The following code examples show how to generate an employee report from an MDB data source by using Office Automation and DocIO.

Using Microsoft Office Automation

Office Automation performs the Mail merge by executing a SQL query on the Word document. The output of the Mail merge can be sent to a new Word document. Alternatively, it can be sent to a printer, a fax machine, or forwarded to an e-mail address.

C#

```
using word = Microsoft.Office.Interop.Word;
-----
//Initializes objects.
object nullobject = Missing.Value;
object filepath = "Sample.docx";
object sqlStmt = "SELECT * FROM [Employees]";
string sDBPath = "Northwind.mdb";
//Starts the Word application.
word.Application wordApp = new word.Application();
//Opens the Word document.
word.Document document = wordApp.Documents.Open(ref filepath, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject);
wordApp.Visible = false;
//Performs Mail Merge.
document.MailMerge.OpenDataSource(sDBPath, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref sqlStmt, ref nullobject,
ref nullobject, ref nullobject);
document.MailMerge.Execute(ref nullobject);
//Sends output of Mail Merge to a new document.
document.MailMerge.Destination =
word.WdMailMergeDestination.wdSendToNewDocument;
//Closes the document.
document.Close(ref nullobject, ref nullobject, ref nullobject);
//Quits the application.
wordApp.Quit(ref nullobject, ref nullobject, ref nullobject);
```

VB.NET

```
Imports word = Microsoft.Office.Interop.Word
-----
'Initializes objects.
Dim nullobject As Object = Missing.Value
Dim filepath As Object = "Sample.docx"
Dim sqlStmt As Object = "SELECT * FROM [Employees]"
Dim sDBPath As String = "Northwind.mdb"
'Starts the Word application.
Dim wordApp As New word.Application()
```

```

'Opens the Word document.
Dim document As Word.Document = wordApp.Documents.Open(filepath, nullobject,
nullobject, nullobject, nullobject, nullobject, _
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject, _
nullobject, nullobject, nullobject, nullobject)
wordApp.Visible = False
'Performs Mail Merge.
document.MailMerge.OpenDataSource(sDBPath, nullobject, nullobject,
nullobject, nullobject, nullobject, _
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject, _
sqlStmt, nullobject, nullobject, nullobject)
document.MailMerge.Execute(nullobject)
'Sends output of Mail Merge to a new document.
document.MailMerge.Destination =
word.WdMailMergeDestination.wdSendToNewDocument
'Closes the document.
document.Close(nullobject, nullobject, nullobject)
'Quits the application.
wordApp.Quit(nullobject, nullobject, nullobject)

```

Using DocIO

DocIO performs Mail merge by using the following methods:

- Execute
- ExecuteGroup
- ExecuteNestedGroup

The following code example performs Mail merge by using the **Execute** method.

C#

```

string dataBase = "Northwind.mdb";
//Opens existing template.
WordDocument doc = new WordDocument("Template.docx", FormatType.Docx);
//Gets Data from the Database.
OleDbConnection conn = new
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + dataBase);
conn.Open();
//Populates the data table.
DataTable table = new DataTable();
OleDbDataAdapter adapter = new OleDbDataAdapter("select * from employees",
conn);
adapter.Fill(table);
adapter.Dispose();
//Performs Mail Merge.
doc.MailMerge.Execute(table);
//Saves the document.
doc.Save("Sample.docx", FormatType.Docx);
//Closes the document.
doc.Close();

```

VB.NET

```

Dim dataBase As String = "Northwind.mdb"

```

```

'Opens the Word document.
Dim doc As WordDocument = New WordDocument("Template.docx")
'Creates database connection.
Dim conn As OleDbConnection = New
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + dataBase)
conn.Open()
'Populates data table.
Dim table As DataTable = New DataTable()
Dim adapter As OleDbDataAdapter = New OleDbDataAdapter("select * from
employees", conn)
adapter.Fill(table)
adapter.Dispose()
'Performs Mail Merge.
doc.MailMerge.Execute(table)
'Saves the document.
doc.Save("Sample.docx", FormatType.Docx)
'Closes the document.
doc.Close()

```

Note:

For more information on Mail merge using DocIO, you can refer to online documentation link:

[MailMerge](#)

Find and Replace

This section illustrates how to perform a simple find and replace operation in a Word document by using Microsoft Office Automation and DocIO.

Using Microsoft Office Automation

The following code example illustrates how to search for a word in a Word document, replace it with another word and save the document under a new name.

C#

```

using word = Microsoft.Office.Interop.Word;
-----
//Initializes objects.
object nullobject = Missing.Value;
object filepath = "Template.docx";
object newFilePath = "Sample.docx";
object item = word.WdGoToItem.wdGoToPage;
object whichItem = word.WdGoToDirection.wdGoToFirst;
object replaceAll = word.WdReplace.wdReplaceAll;
object forward = true;
object matchAllWord = true;
object matchCase = false;
object originalText = "Hello";
object replaceText = "World";
object save = true;
//Starts the Word application.
word.Application wordApp = new word.Application();
//Opens the Word document.
word.Document document = wordApp.Documents.Open(ref filepath, ref
nullobject, ref nullobject,

```

```

ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject,
ref nullobject);
wordApp.Visible = false;
//Searches and replaces text.
document.GoTo(ref item, ref whichItem, ref nullobject, ref nullobject);
foreach (word.Range rng in document.StoryRanges)
{
    rng.Find.Execute(ref originalText, ref matchCase, ref matchAllWord, ref
    nullobject, ref nullobject,
    ref nullobject, ref forward, ref nullobject, ref nullobject, ref replaceText,
    ref replaceAll,
    ref nullobject, ref nullobject, ref nullobject, ref nullobject);
}
//Saves the document.
document.SaveAs(ref newFilePath, ref nullobject, ref nullobject, ref
nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject);
//Closes the document.
document.Close(ref nullobject, ref nullobject, ref nullobject);
//Quits the application.
wordApp.Quit(ref nullobject, ref nullobject, ref nullobject);

```

VB.NET

```

Imports word = Microsoft.Office.Interop.Word
-----
'Initializes objects.
Dim nullobject As Object = Missing.Value
Dim filePath As Object = "Template.docx"
Dim newFilePath As Object = "Sample.docx"
Dim item As Object = word.WdGoToItem.wdGoToPage
Dim whichItem As Object = word.WdGoToDirection.wdGoToFirst
Dim replaceAll As Object = word.WdReplace.wdReplaceAll
Dim forward As Object = True
Dim matchAllWord As Object = True
Dim matchCase As Object = False
Dim originalText As Object = "Hello"
Dim replaceText As Object = "World"
Dim save As Object = True
Dim falseObj As Object = False
'Starts the Word application.
Dim wordApp As word.Application = New word.Application()
'Opens the Word document.
Dim doc As word.Document = wordApp.Documents.Open(filePath, nullobject,
nullobject, nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, falseObj, nullobject, nullobject,
nullobject, nullobject)
wordApp.Visible = False
'Searches and replaces text.
doc.GoTo(item, whichItem, nullobject, nullobject)

```

```

For Each rng As word.Range In doc.StoryRanges
rng.Find.Execute(originalText, matchCase, matchAllWord, nullobject,
nullobject, nullobject, forward, nullobject, nullobject, replaceText,
replaceAll, nullobject, nullobject, nullobject, nullobject)
Next
`Saves the document.
doc.SaveAs(newFilePath, nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, nullobject, nullobject)
`Closes the document.
doc.Close(nullobject, nullobject, nullobject)
`Quits the application.
wordApp.Quit(nullobject, nullobject, nullobject)

```

Using DocIO

The following code example illustrates how to perform a simple find and replace operation by using DocIO.

C#

```

//Opens the Word document.
WordDocument document = new WordDocument("Template.docx", FormatType.Docx);
//Defines replacement text.
string replaceText = "World";
//Performs replace.
document.Replace(new Regex("Hello"), replaceText);
//Saves the document.
document.Save("Sample.docx", FormatType.Docx);
//Closes the document.
document.Close();

```

VB.NET

```

`Opens the Word document.
Dim document As WordDocument = New WordDocument("Template.docx")
`Defines text to be replaced.
Dim replaceText As String = "World"
`Performs replace.
document.Replace(New Regex("Hello"), replaceText)
`Saves the document.
document.Save("Sample.docx", FormatType.Docx)
`Closes the document.
document.Close()

```

Note: For more information on performing the find and replace operation using DocIO, you can refer to online documentation link:

[Find and Replace](#)

Bookmarks

Bookmarks identify the location of text in a Word document that you can name and identify for future reference.

Using Microsoft Office Automation

The following code example illustrates how to insert a bookmark for a range of text by using Office Automation.

C#

```
using word = Microsoft.Office.Interop.Word;
-----
//Initializes objects.
object nullobject = Missing.Value;
object newFilePath = "Sample.docx";
//Starts a Word application.
Microsoft.Office.Interop.Word.Application wordApp = new
Microsoft.Office.Interop.Word.Application();
//Creates a new Word document.
wordApp.Documents.Add(ref nullobject, ref nullobject, ref nullobject, ref
nullobject);
Microsoft.Office.Interop.Word.Document document = wordApp.ActiveDocument;
//Adds a paragraph to the document.
Microsoft.Office.Interop.Word.Paragraph oPara1;
oPara1 = document.Content.Paragraphs.Add(ref nullobject);
oPara1.Range.Text = "Bookmark with one word selected";
//Defines start and end positions of bookmark range.
object start = oPara1.Range.Text.IndexOf("word");
object end = oPara1.Range.Text.LastIndexOf(" ");
object rng = document.Range(ref start, ref end);
//Adds bookmark.
document.Bookmarks.Add("one_word", ref rng);
//Saves document and quits application.
document.SaveAs(ref newFilePath, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject);
//Closes document.
document.Close(ref nullobject, ref nullobject, ref nullobject);
//Quits application.
wordApp.Quit(ref nullobject, ref nullobject, ref nullobject);
```

VB.NET

```
Imports word = Microsoft.Office.Interop.Word
-----
'Initializes objects.
Dim nullobject As Object = Missing.Value
Dim newFilePath As Object = "Sample.docx"
'Starts a Word application.
Dim wordApp As word.Application = New word.Application()
'Creates a new Word document.
wordApp.Documents.Add(nullobject, nullobject, nullobject, nullobject)
Dim doc As word.Document = wordApp.ActiveDocument
'Adds a paragraph to the document.
Dim oPara As word.Paragraph
oPara = doc.Content.Paragraphs.Add(nullobject)
oPara.Range.Text = "Bookmark with one word selected"
'Defines the start and end positions of bookmark range.
Dim startobj As Object = oPara.Range.Text.IndexOf("word")
```



```

Dim endobj As Object = oPara.Range.Text.LastIndexOf(" ")
Dim rng As Object = doc.Range(startobj, endobj)
'Adds bookmark.
doc.Bookmarks.Add("one_word", rng)
'Saves document.
doc.SaveAs(newFilePath)
'Closes document.
doc.Close(nullobject, nullobject, nullobject)
'Quits application.
wordApp.Quit()

```

Using DocIO

The following code example illustrates how to insert the bookmark by using DocIO. Here, the `AppendBookmarkStart()` and `AppendBookmarkEnd()` methods are used to add the bookmark.

C#

```

//Creates a new Word document.
WordDocument doc = new WordDocument();
//Adds new section
IWSection section = doc.AddSection();
//Adds new paragraph
IWParagraph paragraph = section.AddParagraph();
paragraph.AppendText("Simple Bookmark");
paragraph = section.AddParagraph();
paragraph.AppendText("Bookmark with one ");
//Inserts bookmark.
paragraph.AppendBookmarkStart("one_word");
paragraph.AppendText("word");
paragraph.AppendBookmarkEnd("one_word");
paragraph.AppendText(" selected");
//Saves the document.
doc.Save("Sample.docx", FormatType.Docx);
//Closes the document.
doc.Close();

```

VB.NET

```

'Creates a new Word document.
Dim doc As WordDocument = New WordDocument()
'Adds new section
Dim section As IWSection = doc.AddSection()
'Adds new paragraph
Dim paragraph As IWParagraph = section.AddParagraph()
paragraph.AppendText("Simple Bookmark")
paragraph = section.AddParagraph()
paragraph.AppendText("Bookmark with one ")
'Inserts bookmark.
paragraph.AppendBookmarkStart("one_word")
paragraph.AppendText("word")
paragraph.AppendBookmarkEnd("one_word")
paragraph.AppendText(" selected")
'Saves the document.
doc.Save("Sample.docx", FormatType.Docx)
'Closes the document.

```

```
doc.Close()
```

Page Numbers

Page numbers can be added to the Word document in headers or footers.

Using Microsoft Office Automation

The following code example illustrates how page numbers can be inserted to the footer of the Word document by adding a page number field.

C#

```
using word = Microsoft.Office.Interop.Word;
-----
//Initializes objects.
object filepath = "Sample.docx";
object nullobject = Missing.Value;
//Starts the Word application.
word.Application wordApp = new word.Application();
//Opens the Word document.
word.Document document = wordApp.Documents.Open(ref filepath, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject);
wordApp.Visible = false;
document.Activate();
//Seeks the page footer.
wordApp.ActiveWindow.ActivePane.View.SeekView =
Microsoft.Office.Interop.Word.WdSeekView.wdSeekCurrentPageFooter;
//Formats the footer.
wordApp.Selection.Paragraphs.Alignment =
word.WdParagraphAlignment.wdAlignParagraphCenter;
wordApp.ActiveWindow.Selection.Font.Name = "Arial";
wordApp.ActiveWindow.Selection.Font.Size = 8;
//Adds page numbers in the footer.
Object CurrentPage = word.WdFieldType.wdFieldPage;
wordApp.ActiveWindow.Selection.Fields.Add(wordApp.Selection.Range, ref
CurrentPage, ref nullobject, ref nullobject);
//Saves the document.
document.Save();
//Closes the document.
document.Close(ref nullobject, ref nullobject, ref nullobject);
//Quits the application.
wordApp.Quit(ref nullobject, ref nullobject, ref nullobject);
```

VB.NET

```
Imports word = Microsoft.Office.Interop.Word
-----
'Initializes objects.
Dim nullobject As Object = Missing.Value
Dim filePath As Object = "Sample.docx"
Dim falseobj As Object = False
'Starts the application.
Dim wordApp As word.Application = New word.Application()
```

```

`Adds a new Word document.
Dim document As word.Document = wordApp.Documents.Open(filePath, nullobject,
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, falseobj, nullobject, nullobject,
nullobject, nullobject)
wordApp.Visible = False
document.Activate()
`Seeks the page footer.
wordApp.ActiveWindow.ActivePane.View.SeekView =
word.WdSeekView.wdSeekCurrentPageFooter
`Formats the footer.
wordApp.Selection.Paragraphs.Alignment =
word.WdParagraphAlignment.wdAlignParagraphCenter
wordApp.ActiveWindow.Selection.Font.Name = "Arial"
wordApp.ActiveWindow.Selection.Font.Size = 8
`Adds page numbers in the footer.
Dim CurrentPage As Object = word.WdFieldType.wdFieldPage
wordApp.ActiveWindow.Selection.Fields.Add(wordApp.Selection.Range,
CurrentPage, nullobject, nullobject)
`Saves the document.
document.Save()
`Closes the document.
document.Close(nullobject, nullobject, nullobject)
`Quits application.
wordApp.Quit()

```

Using DocIO

The following code example illustrates how page numbers are inserted to the footer of the Word document by using DocIO.

C#

```

//Opens the Word document.
WordDocument doc = new WordDocument("Template.docx", FormatType.Docx);
//Iterates through sections
foreach (WSection sec in doc.Sections)
{
    IWParagraph para = sec.AddParagraph();
    //Appends page field to the paragraph
    para.AppendField("footer", FieldType.FieldPage);
    para.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Center;
    sec.PageSetup.PageNumberStyle = PageNumberStyle.Arabic;
    //Adds paragraph to footer
    sec.HeadersFooters.Footer.Paragraphs.Add(para);
}
//Saves the document.
doc.Save("Sample.docx", FormatType.Docx);
//Closes the document.
doc.Close();

```

VB.NET

```

`Opens the Word document.
Dim doc As WordDocument = New WordDocument("Template.docx", FormatType.Docx)
`Iterates through sections

```

```

For Each sec As WSection In doc.Sections
Dim para As IWParagraph = sec.AddParagraph()
'Appends page field to the paragraph
para.AppendField("footer", FieldType.FieldPage)
para.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Center
sec.PageSetup.PageNumberStyle = PageNumberStyle.Arabic
'Adds paragraph to footer
sec.HeadersFooters.Footer.Paragraphs.Add(para)
Next
'Saves the document.
doc.Save("Sample.docx", FormatType.Docx)
'Closes the document.
doc.Close()

```

Document Watermarks

Watermarks are text or pictures that appear behind document text.

Using Microsoft Office Automation

The following code example illustrates how to insert a text watermark as a shape by using Office Automation.

C#

```

using word = Microsoft.Office.Interop.Word;
-----
//Initializes objects.
object nullobject = Missing.Value;
object newFilePath = "Sample.docx";
//Starts the Word application.
word.Application wordApp = new word.Application();
//Creates a new Word document.
wordApp.Documents.Add(ref nullobject, ref nullobject, ref nullobject, ref
nullobject);
word.Document document = wordApp.ActiveDocument;
//Seeks the current page header.
wordApp.ActiveWindow.ActivePane.View.SeekView =
word.WdSeekView.wdSeekCurrentPageHeader;
//Inserts watermark.
word.Shape watermark =
wordApp.Selection.HeaderFooter.Shapes.AddTextEffect(Microsoft.Office.Core.Ms
oPresetTextEffect.msoTextEffect1,
"Watermark", "Arial", (float)48, Microsoft.Office.Core.MsoTriState.msoTrue,
Microsoft.Office.Core.MsoTriState.msoFalse, 0, 0, ref nullobject);
//Sets watermark properties.
watermark.Fill.Visible = Microsoft.Office.Core.MsoTriState.msoTrue;
watermark.Line.Visible = Microsoft.Office.Core.MsoTriState.msoFalse;
watermark.Fill.Solid();
watermark.Fill.ForeColor.RGB = (Int32)word.WdColor.wdColorGray30;
//Sets focus back to the document.
wordApp.ActiveWindow.ActivePane.View.SeekView =
word.WdSeekView.wdSeekMainDocument;
//Saves the document.
document.SaveAs(ref newFilePath, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject,

```

```

ref nullobject);
//Closes the document.
document.Close(ref nullobject, ref nullobject, ref nullobject);
//Quits the application.
wordApp.Quit(ref nullobject, ref nullobject, ref nullobject);

```

VB.NET

```

Imports word = Microsoft.Office.Interop.Word
-----
Initializes objects.
Dim nullobject As Object = Missing.Value
Dim newFilePath As Object = "Sample.docx"
`Starts the application.
Dim wordApp As word.Application = New word.Application()
`Creates a new Word document.
wordApp.Documents.Add(nullobject, nullobject, nullobject, nullobject)
Dim doc As word.Document = wordApp.ActiveDocument
`Seeks the current page header.
wordApp.ActiveWindow.ActivePane.View.SeekView =
word.WdSeekView.wdSeekCurrentPageHeader
`Adds text watermark to the document.
Dim watermark As word.Shape =
wordApp.Selection.HeaderFooter.Shapes.AddTextEffect(Microsoft.Office.Core.Ms
oPresetTextEffect.msoTextEffect1,"Watermark", "Arial", 48,
Microsoft.Office.Core.MsoTriState.msoTrue,
Microsoft.Office.Core.MsoTriState.msoFalse, 0, 0, nullobject)
`Sets watermark properties.
watermark.Fill.Visible = Microsoft.Office.Core.MsoTriState.msoTrue
watermark.Line.Visible = Microsoft.Office.Core.MsoTriState.msoFalse
watermark.Fill.Solid()
watermark.Fill.ForeColor.RGB = CType(word.WdColor.wdColorGray30, Integer)
`Saves the document.
doc.SaveAs(newFilePath, nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, nullobject, nullobject)
`Closes the document.
doc.Close(nullobject, nullobject, nullobject)
`Quits application.
wordApp.Quit()

```

Using DocIO

DocIO enables you to add a text watermark and a picture watermark to a Word document. The following code example shows how to insert the picture watermark to the Word document.

C#

```

//Creates a new Word document.
WordDocument doc = new WordDocument();
doc.EnsureMinimal();
//Adds picture watermark to the document.
PictureWatermark picWatermark = new PictureWatermark();
picWatermark.Scaling = 120f;
picWatermark.Washout = true;
doc.Watermark = picWatermark;

```

```

picWatermark.Picture = Image.FromFile(ImagesPath + "Water lilies.jpg");
//Saves the document.
doc.Save("Sample.docx", FormatType.Docx);
//Closes the document.
doc.Close();

```

VB.NET

```

'Creates a new Word document.
Dim doc As WordDocument = New WordDocument()
doc.EnsureMinimal()
'Adds picture watermark to the document.
Dim picWatermark As PictureWatermark = New PictureWatermark()
picWatermark.Scaling = 120f
picWatermark.Washout = True
doc.Watermark = picWatermark
picWatermark.Picture = Image.FromFile(ImagesPath and "Water lilies.jpg")
'Saves the document.
doc.Save("Sample.docx", FormatType.Docx)
'Closes the document.
doc.Close()

```

Note: For more information on adding watermarks to a Word document using DocIO, refer to the online documentation link:

[Applying Watermark](#)

Headers and Footers

The headers and footers can be inserted with text, graphics, and any other information that is contained in the document.

Using Microsoft Office Automation

The following code example illustrates how to add headers and footers to a Word document. In this example, page numbers are inserted to the header and a text is inserted to the footer.

C#

```

using word = Microsoft.Office.Interop.Word;
-----
//Initializes objects.
object nullobject = Missing.Value;
object filePath = "Template.docx";
object newFilePath = "Sample.docx";
//Starts the Word application.
word.Application wordApp = new word.Application();
//Opens the Word document.
word.Document document = wordApp.Documents.Open(ref filePath, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject);
wordApp.Visible = false;
//Adds header and footer to each section in the document.
foreach (word.Section section in document.Sections)
{

```

```

object fieldEmpty = word.WdFieldType.wdFieldPage;
object autoText = "AUTOTEXT \\"Page X of Y\\" ";
object preserveFormatting = true;
//Footer.
section.Footers[word.WdHeaderFooterIndex.wdHeaderFooterPrimary].Range.Text =
"Internal";
section.Footers[word.WdHeaderFooterIndex.wdHeaderFooterPrimary].Range.ParagraphFormat.Alignment =
word.WdParagraphAlignment.wdAlignParagraphLeft;
//Header.
section.Headers[word.WdHeaderFooterIndex.wdHeaderFooterPrimary].Range.Fields
.Add(section.Headers[
word.WdHeaderFooterIndex.wdHeaderFooterPrimary].Range,reffieldEmpty, ref
autoText, ref preserveFormatting);
section.Headers[word.WdHeaderFooterIndex.wdHeaderFooterPrimary].Range.ParagraphFormat.Alignment =
word.WdParagraphAlignment.wdAlignParagraphRight;
}
//Saves the document.
document.SaveAs(ref newFilePath, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject);
//Closes the document.
document.Close(ref nullobject, ref nullobject, ref nullobject);
//Quits the application.
wordApp.Quit(ref nullobject, ref nullobject, ref nullobject);

```

VB.NET

```

Imports word = Microsoft.Office.Interop.Word
-----
'Initializes objects.
Dim nullobject As Object = System.Reflection.Missing.Value
Dim filePath As Object = "Template.docx"
Dim newFilePath As Object = "Sample.docx"
'Starts the application.
Dim wordApp As word.Application = New word.Application()
'Opens the document.
Dim document As word.Document = wordApp.Documents.Open(filePath, nullobject,
nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, nullobject)
wordApp.Visible = False
'Adds header and footer to each section in the document.
For Each section As word.Section In document.Sections
Dim fieldEmpty As Object = word.WdFieldType.wdFieldPage
'Footer.
section.Footers(word.WdHeaderFooterIndex.wdHeaderFooterPrimary).Range.Text =
"Internal"
section.Footers(word.WdHeaderFooterIndex.wdHeaderFooterPrimary).Range.ParagraphFormat.Alignment = word.WdParagraphAlignment.wdAlignParagraphLeft
'Header.

```

```

section.Headers(word.WdHeaderFooterIndex.wdHeaderFooterPrimary).Range.Fields
.Add(section.Headers(word.WdHeaderFooterIndex.wdHeaderFooterPrimary).Range,
fieldEmpty, nullobject, nullobject)
section.Headers(word.WdHeaderFooterIndex.wdHeaderFooterPrimary).Range.Paragr
aphFormat.Alignment = word.WdParagraphAlignment.wdAlignParagraphRight
Next
'Saves the document.
document.SaveAs(newFilePath, nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, nullobject, nullobject)
'Closes the document.
document.Close(nullobject, nullobject, nullobject)
'Quits the application.
wordApp.Quit()

```

Using DocIO

You can set the header and footer by using the HeadersFooters property in the Word document section. To access a particular header/footer, you can use the following properties of **WHeadersFooters** class:

- FirstPageHeader
- FirstPageFooter
- OddHeader
- OddFooter
- EvenHeader
- EvenFooter

C#

```

//Opens a Word document.
WordDocument doc = new WordDocument("Template.docx");
//Adds header and footer to each section in the document.
foreach (WSection sec in doc.Sections)
{
    //Header.
    WParagraph para = new WParagraph(doc);
    para.AppendField("page", FieldType.FieldPage);
    para.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
    sec.HeadersFooters.Header.Paragraphs.Add(para);
    //Footer.
    WParagraph para1 = new WParagraph(doc);
    para1.AppendText("Internal");
    para1.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Left;
    sec.HeadersFooters.Footer.Paragraphs.Add(para1);
}
//Saves the document.
doc.Save("Sample.docx", FormatType.Docx);
//Closes the document.
doc.Close();

```

VB.NET

```

'Opens the Word document.
Dim doc As WordDocument = New WordDocument("Template.docx")

```



```

`Adds header and footer to each section in the document.
For Each sec As WSection In doc.Sections
`Header.
Dim para As WParagraph = New WParagraph(doc)
para.AppendField("page", FieldType.FieldPage)
para.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right
sec.HeadersFooters.Header.Paragraphs.Add(para)
`Footer.
Dim para1 As WParagraph = New WParagraph(doc)
para1.AppendText("Internal")
para1.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Left
sec.HeadersFooters.Footer.Paragraphs.Add(para1)
Next
`Saves the document.
doc.Save("Sample.docx", FormatType.Docx)
`Closes the document.
doc.Close()

```

Character Formatting

Character formatting defines the appearance of the text in a Word document. This section illustrates how to apply character level formatting to the Word document.

Using Microsoft Office Automation

The following code example illustrates how to apply the character formatting to the Word document by using the Range properties.

C#

```

using word = Microsoft.Office.Interop.Word
-----
//Initializes objects.
object nullobject = System.Reflection.Missing.Value;
object newFilePath = "Sample.docx";
object falseObj = false;
//Starts the Word application.
word.Application wordApp = new word.Application();
//Creates a new Word document.
wordApp.Documents.Add(ref nullobject, ref nullobject, ref nullobject, ref
nullobject);
word.Document doc = wordApp.ActiveDocument;
//Defines the range for formatting.
object start = 0;
object end = 0;
word.Range rng = doc.Range(ref start, ref end);
rng.Text = "New Text";
rng.Font.Name = "Arial";
rng.Font.Size = 14;
//Saves the document.
doc.SaveAs(ref newFilePath, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject);
//Closes the document.
doc.Close(ref nullobject, ref nullobject, ref nullobject);

```

```
//Quits the application.
wordApp.Quit(ref nullobject, ref nullobject, ref nullobject);
```

VB.NET

```
Imports word = Microsoft.Office.Interop.Word
-----
'Initializes objects.
Dim nullobject As Object = System.Reflection.Missing.Value
Dim newFilePath As Object = "Sample.docx"
Dim falseObj As Object = False
'Starts the Word application.
Dim wordApp As word.Application = New word.Application()
'Creates a new Word document.
wordApp.Documents.Add(nullobject, nullobject, nullobject, nullobject)
Dim doc As word.Document = wordApp.ActiveDocument
'Defines the range for formatting.
Dim start As Object = 0
Dim endobj As Object = 0
Dim rng As word.Range = doc.Range(start, endobj)
rng.Text = "New Text"
rng.Font.Name = "Arial"
rng.Font.Size = 14
'Saves the document.
doc.SaveAs(newFilePath, nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, nullobject, nullobject)
'Closes the document.
doc.Close(nullobject, nullobject, nullobject)
'Quits the application.
wordApp.Quit()
```

Tables

Tables are used to organize information and to display the information in rows and columns. You can also add images or even other tables to the table.

Using Microsoft Office Automation

The following code example illustrates how to insert a table to a Word document, where the table contains three rows and two columns.

C#

```
using word = Microsoft.Office.Interop.Word;
-----
//Initializes the objects.
object nullobject = System.Reflection.Missing.Value;
object newFilePath = "Sample.docx";
//Starts the Word application.
word.Application wordApp = new word.Application();
//Creates a new document.
wordApp.Documents.Add(ref nullobject, ref nullobject, ref nullobject, ref
nullobject);
word.Document document = wordApp.ActiveDocument;
//Inserts the table.
object start = 0;
```

```

object end = 0;
word.Range tableLocation = document.Range(ref start, ref end);
document.Tables.Add(tableLocation, 3, 2, ref nullobject, ref nullobject);
//Saves the document.
document.SaveAs(ref newFilePath, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref
ref nullobject);
//Closes the document.
document.Close(ref nullobject, ref nullobject, ref nullobject);
//Quits the application.
wordApp.Quit(ref nullobject, ref nullobject, ref nullobject);

```

VB.NET

```

Imports word = Microsoft.Office.Interop.Word
-----
'Initializes the objects.
Dim nullobject As Object = System.Reflection.Missing.Value
Dim newFilePath As Object = "Sample.docx"
'Starts the Word application.
Dim wordApp As New word.Application()
'Creates a new document.
wordApp.Documents.Add(nullobject, nullobject, nullobject, nullobject)
Dim document As word.Document = wordApp.ActiveDocument
'Inserts the table.
Dim start As Object = 0
Dim [end] As Object = 0
Dim tableLocation As word.Range = document.Range(start, [end])
document.Tables.Add(tableLocation, 3, 2, nullobject, nullobject)
'Saves the document.
document.SaveAs(newFilePath, nullobject, nullobject, nullobject, nullobject,
nullobject, _
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject, _
nullobject, nullobject, nullobject, nullobject)
'Closes the document.
document.Close(nullobject, nullobject, nullobject)
'Quits the application.
wordApp.Quit(nullobject, nullobject, nullobject)

```

Using DocIO

The following code example shows how to insert an empty table to a Word document. The `ResetCells()` method is used to specify the number of rows and columns in a table.

C#

```

//Creates a new Word document.
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
//Adds a table to the document.
IWTable table = section.AddTable();
table.ResetCells(3, 2);
//Saves the document.
document.Save("Sample.docx", FormatType.Docx);

```

```
//Closes the document.
document.Close();
```

VB.NET

```
'Creates a new Word document.
Dim document As New WordDocument()
Dim section As IWSection = document.AddSection()
'Adds a table to the document.
Dim table As IWTable = section.AddTable()
table.ResetCells(3, 2)
'Saves the document.
document.Save("Sample.docx", FormatType.Docx);
'Closes the document.
document.Close()
```

Note: For more information on creating tables using DocIO, refer to online documentation link:

[Working with Tables](#)

Comments

Comments are used to include additional information to a paragraph or text in a Word document. Comments can be added or modified whenever needed and deleted when the comment has served its purpose.

Adding Comments using Microsoft Office Automation

The following code example illustrates how to add comments to a Word document. You need to define the range of text where the comment is to be added.

C#

```
using word = Microsoft.Office.Interop.Word;
-----
//Initializes objects.
object nullobject = System.Reflection.Missing.Value;
object newFilePath = "Sample.docx";
//Starts the Word application.
word.Application wordApp = new word.Application();
//Creates a new document.
wordApp.Documents.Add(ref nullobject, ref nullobject, ref nullobject, ref
nullobject);
word.Document doc = wordApp.ActiveDocument;
//Inserts text to the Word document.
object start = 0;
object end = 0;
word.Range rng = doc.Range(ref start, ref end);
rng.Text = "New Text";
//Adds comment to the inserted text.
object text = "Comment goes here";
doc.Comments.Add(rng, ref text);
//Saves the document.
doc.SaveAs(ref newFilePath, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
```

```

ref nullobject);
//Closes the document.
doc.Close(ref nullobject, ref nullobject, ref nullobject);
//Quits the application.
wordApp.Quit(ref nullobject, ref nullobject, ref nullobject);

```

VB.NET

```

Imports word = Microsoft.Office.Interop.Word
-----
'Initializes objects.
Dim nullobject As Object = System.Reflection.Missing.Value
Dim newFilePath As Object = "Sample.docx"
'Starts the Word application.
Dim wordApp As word.Application = New word.Application()
'Creates a new document.
wordApp.Documents.Add(nullobject, nullobject, nullobject, nullobject)
Dim doc As word.Document = wordApp.ActiveDocument
'Inserts text to the Word document.
Dim startobj As Object = 0
Dim endobj As Object = 0
Dim rng As word.Range = doc.Range(startobj, endobj)
rng.Text = "New Text"
'Adds comment to the inserted text.
Dim text As Object = "Comment goes here"
doc.Comments.Add(rng, text)
'Saves the document and quits application.
doc.SaveAs(newFilePath, nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject,
nullobject, nullobject, nullobject, nullobject, nullobject)
'Closes the document.
doc.Close(nullobject, nullobject, nullobject)
'Quits the application.
wordApp.Quit()

```

Adding Comments Using DocIO

You can insert comments to a paragraph or text in a Word document by using DocIO. The following code example shows how to insert comments to a Word document.

C#

```

//Creates a new Word document.
WordDocument doc = new WordDocument();
IWSection section = doc.AddSection();
//Adds a paragraph to the document.
IWParagraph para = section.AddParagraph();
para.AppendText("New Text");
//Adds comment to the paragraph.
para.AppendComment("Comment goes here");
//Saves the document.
doc.Save("Sample.doc", FormatType.Doc);

```

VB.NET

```

'Creates a new Word document.

```

```

Dim doc As WordDocument = New WordDocument()
Dim section As IWSection = doc.AddSection()
'Adds a paragraph to the document.
Dim para As IWParagraph = section.AddParagraph()
para.AppendText("New Text")
para.AppendComment("Comment goes here")
'Saves the document.
doc.Save("Sample.doc", FormatType.Doc)

```

Note: For more information on working with the comments using DocIO, you can refer to the online documentation link:

[Working with Comments](#)

Document Protection

You can protect your Word documents with or without a password from anyone accidentally or deliberately modifying the Word documents. You can specify the protection type for preserving the Word documents.

Using Microsoft Office Automation

[WdProtectionType](#) is used to specify the protection type of the Word document.

C#

```

//Initializes objects.
object nullobject = System.Reflection.Missing.Value;
object filepath = "Template.docx";
object newFilePath = "Sample.docx";
object noReset = false;
object password = System.String.Empty;
object useIRM = false;
object enforceStyleLock = false;
//Starts the Word application.
word.Application wordApp = new word.Application();
//Opens the Word document to be protected.
word.Document document = wordApp.Documents.Open(ref filepath, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject);
wordApp.Visible = false;
//Sets "Allow only Comments" protection to Word document.
document.Protect(word.WdProtectionType.wdAllowOnlyComments, ref noReset, ref
password, ref useIRM, ref enforceStyleLock);
//Saves the document.
document.SaveAs(ref newFilePath, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject);
//Closes the document.
document.Close(ref nullobject, ref nullobject, ref nullobject);
//Quits the application.
wordApp.Quit(ref nullobject, ref nullobject, ref nullobject);

```

VB.NET

```

'Initializes objects.
Dim nullobject As Object = System.Reflection.Missing.Value
Dim filepath As Object = "Template.docx"
Dim newFilePath As Object = "Sample.docx"
Dim noReset As Object = False
Dim password As Object = System.[String].Empty
Dim useIRM As Object = False
Dim enforceStyleLock As Object = False
'Starts the Word application.
Dim wordApp As New Word.Application()
'Opens the Word document that is to be protected.
Dim document As Word.Document = wordApp.Documents.Open(filepath, nullobject,
nullobject, nullobject, nullobject, nullobject, _
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject, _
nullobject, nullobject, nullobject, nullobject)
wordApp.Visible = False
'Sets "Allow only Comments" protection to Word document.
document.Protect(Word.WdProtectionType.wdAllowOnlyComments, noReset,
password, useIRM, enforceStyleLock)
'Saves the document.
document.SaveAs(newFilePath, nullobject, nullobject, nullobject, nullobject,
nullobject, _
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject, _
nullobject, nullobject, nullobject, nullobject)
'Closes the document.
document.Close(nullobject, nullobject, nullobject)
'Quits the application.
wordApp.Quit(nullobject, nullobject, nullobject)

```

Using DocIO

DocIO uses ProtectionType property to specify the protection type of the Word document. This property uses the following values:

- AllowOnlyComments: Allows only comments to be added to the document.
- AllowOnlyFormFields: Allows content to be added to the document through form fields only.
- AllowOnlyRevisions: Allows only revisions to be made to the existing content.
- AllowOnlyReading: All kinds of editing are restricted here and it makes the Word document as read-only document.
- NoProtection: Does not protect the document.

C#

```

//Loads the existing Word document by using DocIO instance
WordDocument document = new WordDocument("Template.docx", FormatType.Docx);
//Sets "Allow only Comments" protection to Word document.
document.ProtectionType = Syncfusion.DocIO.ProtectionType.AllowOnlyComments;
//Saves and closes the document.
document.Save("Sample.docx", FormatType.Docx);
document.Close();

```

VB.NET

```
'Loads the existing Word document by using DocIO instance
Dim document As New WordDocument("Template.docx", FormatType.Docx)
'Sets "Allow only Comments" protection to Word document.
document.ProtectionType = Syncfusion.DocIO.ProtectionType.AllowOnlyComments
'Saves and closes the document.
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

Refer to the online documentation link for more details about the ways to protect the Word documents by using DocIO:

[Protecting word document from editing](#)

Table of Contents

Table of contents can be generated by applying the heading styles to text in a Word document. To create the table of contents in Microsoft Word, click Table of Contents from the Table of Contents group on the References tab.

Using Microsoft Office Automation

The following code example shows how to insert and update table of contents in a Word document.

C#

```
//Initializes objects.
object nullobject = System.Reflection.Missing.Value;
object filepath = "Template.docx";
object newFilePath = "Sample.docx";
object trueobj = true;
//Starts the Word application.
word.Application wordApp = new word.Application();
//Opens the Word document.
word.Document document = wordApp.Documents.Open(ref filepath, ref
nullobject, ref nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject);
wordApp.Visible = false;
//Defines the range for TOC in the document.
object tocstart = 0;
object tocend = 0;
word.Range rngToc = document.Range(ref tocstart, ref tocend);
//Adds TOC.
word.TableOfContents tableOfContents = document.TablesOfContents.Add(rngToc,
ref trueobj, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref trueobj, ref trueobj, ref trueobj, ref trueobj, ref
trueobj, ref trueobj);
//Updates TOC.
tableOfContents.Update();
//Saves the document.
document.SaveAs(ref newFilePath, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject, ref nullobject, ref nullobject, ref
nullobject, ref nullobject, ref nullobject,
ref nullobject, ref nullobject);
//Closes the document.
document.Close(ref nullobject, ref nullobject, ref nullobject);
```



```
//Quits the application.
wordApp.Quit(ref nullobject, ref nullobject, ref nullobject);
```

VB.NET

```
'Initializes objects.
Dim nullobject As Object = System.Reflection.Missing.Value
Dim filepath As Object = "Template.docx"
Dim newFilePath As Object = "Sample.docx"
Dim trueobj As Object = True
'Starts the Word application.
Dim wordApp As New Word.Application()
'Opens the Word document.
Dim document As Word.Document = wordApp.Documents.Open(filepath, nullobject,
nullobject, nullobject, nullobject, nullobject, _
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject, _
nullobject, nullobject, nullobject, nullobject)
wordApp.Visible = False
'Defines the range for TOC in the document.
Dim tocstart As Object = 0
Dim tocend As Object = 0
Dim rngToc As Word.Range = document.Range(tocstart, tocend)
'Adds TOC.
Dim tableOfContents As Word.TableOfContents =
document.TablesOfContents.Add(rngToc, trueobj, nullobject, nullobject,
nullobject, nullobject, _
trueobj, trueobj, trueobj, trueobj, trueobj, trueobj)
'Updates TOC.
tableOfContents.Update()
'Saves the document.
document.SaveAs(newFilePath, nullobject, nullobject, nullobject, nullobject,
nullobject, _
nullobject, nullobject, nullobject, nullobject, nullobject, nullobject, _
nullobject, nullobject, nullobject, nullobject)
'Closes the document.
document.Close(nullobject, nullobject, nullobject)
'Quits the application.
wordApp.Quit(nullobject, nullobject, nullobject)
```

Using DocIO

The following code example illustrates how to insert and update the table of contents in a Word document by using DocIO.

C#

```
//Loads the existing Word document by using DocIO instance
WordDocument document = new WordDocument("Template.docx", FormatType.Docx);
IWSection section = document.Sections[0];
//Appends TOC to the first paragraph of the document.
WParagraph paragraph = new WParagraph(document);
TableOfContent tableOfContents = paragraph.AppendTOC(1, 3);
section.Paragraphs.Insert(0, paragraph);
//Updates table of contents.
document.UpdateTableOfContents();
//Saves and closes the document.
```

```
document.Save("Sample.docx", FormatType.Docx);
document.Close();
```

VB.NET

```
'Loads the existing Word document by using DocIO instance
Dim document As New WordDocument("Template.docx", FormatType.Docx)
Dim section As IWSection = document.Sections(0)
'Appends TOC to the first paragraph of the document.
Dim paragraph As New WParagraph(document)
Dim tableOfContents As TableOfContent = paragraph.AppendTOC(1, 3)
section.Paragraphs.Insert(0, paragraph)
'Updates table of contents.
document.UpdateTableOfContents()
'Saves and closes the document.
document.Save("Sample.docx", FormatType.Docx)
document.Close()
```

Refer to the online documentation link for more information about adding the table of contents to the Word document by using DocIO:

[Working with table of contents](#)

How to copy necessary fonts to Linux containers

The fonts present in the location(in Docker container) "/usr/local/share/fonts/" is used for conversion. By default, there will be limited number of fonts available in the container.

You should copy necessary fonts to this location "/usr/local/share/fonts/" before conversion.

Use the following code example to copy fonts to containers.

DOCKERFILE

PDF

C#

```
using Syncfusion.Pdf;
using Syncfusion.Pdf.Parsing;
using Syncfusion.Pdf.Graphics;
using Syncfusion.Pdf.Grid;
```

VB.NET

```
Imports Syncfusion.Pdf
Imports Syncfusion.Pdf.Parsing
Imports Syncfusion.Pdf.Graphics
Imports Syncfusion.Pdf.Grid
```

UWP

```
using Syncfusion.Pdf;
using Syncfusion.Pdf.Parsing;
```

```
using Syncfusion.Pdf.Graphics;  
using Syncfusion.Pdf.Grid;
```

ASP.NET CORE

```
using Syncfusion.Pdf;  
using Syncfusion.Pdf.Parsing;  
using Syncfusion.Pdf.Graphics;  
using Syncfusion.Pdf.Grid;
```

XAMARIN

```
using Syncfusion.Pdf;  
using Syncfusion.Pdf.Parsing;  
using Syncfusion.Pdf.Graphics;  
using Syncfusion.Pdf.Grid;
```

Creating a PDF document with simple text

The following code example shows how to create a PDF document with simple text.

C#

```
//Create a new PDF document.  
PdfDocument document = new PdfDocument();  
//Add a page to the document.  
PdfPage page = document.Pages.Add();  
//Create PDF graphics for the page.  
PdfGraphics graphics = page.Graphics;  
//Set the standard font.  
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);  
//Draw the text.  
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,  
0));  
//Save the document.  
document.Save("Output.pdf");  
//Close the document.  
document.Close(true);
```

VB.NET

```
'Create a new PDF document.  
Dim document As New PdfDocument()  
'Add a page to the document.  
Dim page As PdfPage = document.Pages.Add()  
'Create PDF graphics for the page.  
Dim graphics As PdfGraphics = page.Graphics  
'Set the standard font.  
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)  
'Draw the text.  
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0,  
0))  
'Save the document.  
document.Save("Output.pdf")  
'Close the document.  
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
```

```

//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Creating a PDF document with image

The following code example shows how to create a PDF document with an image.

C#

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page to the document.
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image from the disk.
PdfBitmap image = new PdfBitmap("Autumn Leaves.jpg");
//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the document.
doc.Save("Output.pdf");
//Close the document.
doc.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page to the document.
Dim page As PdfPage = doc.Pages.Add()

```

```

'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Load the image from the disk.
Dim image As New PdfBitmap("Autumn Leaves.jpg")
'Draw the image
graphics.DrawImage(image, 0, 0)
'Save the document.
doc.Save("Output.pdf")
'Close the document.
doc.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page to the document.
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image as stream.
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Autumn Leaves.jpg");
PdfBitmap image = new PdfBitmap(imageStream);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page to the document.
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image as stream.
FileStream imageStream = new FileStream("Autumn Leaves.jpg", FileMode.Open,
FileAccess.Read);
PdfBitmap image = new PdfBitmap(imageStream);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);

```

```
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page to the document.
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image as stream.
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Autumn Leaves.jpg");
PdfBitmap image = new PdfBitmap(imageStream);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Creating a PDF document with table

The following code example shows how to create a PDF document with a simple table.

C#

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page
PdfPage page = doc.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable
```

```

DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ProductID");
dataTable.Columns.Add("ProductName");
dataTable.Columns.Add("Quantity");
dataTable.Columns.Add("UnitPrice");
dataTable.Columns.Add("Discount");
dataTable.Columns.Add("Price");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "CA-1098", "Queso Cabrales", "12", "14",
"1", "167" });
dataTable.Rows.Add(new object[] { "LJ-0192-M", "Singaporean Hokkien Fried
Mee", "10", "20", "3", "197" });
dataTable.Rows.Add(new object[] { "SO-B909-M", "Mozzarella di Giovanni",
"15", "65", "10", "956" });
//Assign data source
pdfGrid.DataSource = dataTable;
//Draw grid to the page of PDF document
pdfGrid.Draw(page, new PointF(10, 10));
//Save the document
doc.Save("Output.pdf");
//Close the document
doc.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page.
Dim page As PdfPage = doc.Pages.Add()
'Create a PdfGrid.
Dim pdfGrid As New PdfGrid()
'Create a DataTable.
Dim dataTable As New DataTable()
'Add columns to the DataTable
dataTable.Columns.Add("ProductID")
dataTable.Columns.Add("ProductName")
dataTable.Columns.Add("Quantity")
dataTable.Columns.Add("UnitPrice")
dataTable.Columns.Add("Discount")
dataTable.Columns.Add("Price")
'Add rows to the DataTable
dataTable.Rows.Add(New Object() {"CA-1098", "Queso Cabrales", "12", "14",
"1", "167"})
dataTable.Rows.Add(New Object() {"LJ-0192-M", "Singaporean Hokkien Fried
Mee", "10", "20", "3", "197"})
dataTable.Rows.Add(New Object() {"SO-B909-M", "Mozzarella di Giovanni",
"15", "65", "10", "956"})
'Assign data source
pdfGrid.DataSource = dataTable
'Draw grid to the page of PDF document
pdfGrid.Draw(page, New PointF(10, 10))
'Save the document
doc.Save("Output.pdf")
'Close the document
doc.Close(true)

```


UWP

```

///Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page
PdfPage page = doc.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ProductID");
dataTable.Columns.Add("ProductName");
dataTable.Columns.Add("Quantity");
dataTable.Columns.Add("UnitPrice");
dataTable.Columns.Add("Discount");
dataTable.Columns.Add("Price");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "CA-1098", "Queso Cabrales", "12", "14",
"1", "167" });
dataTable.Rows.Add(new object[] { "LJ-0192-M", "Singaporean Hokkien Fried
Mee", "10", "20", "3", "197" });
dataTable.Rows.Add(new object[] { "SO-B909-M", "Mozzarella di Giovanni",
"15", "65", "10", "956" });
//Assign data source
pdfGrid.DataSource = dataTable;
//Draw grid to the page of PDF document
pdfGrid.Draw(page, new PointF(10, 10));
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page
PdfPage page = doc.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ProductID");
dataTable.Columns.Add("ProductName");
dataTable.Columns.Add("Quantity");
dataTable.Columns.Add("UnitPrice");
dataTable.Columns.Add("Discount");
dataTable.Columns.Add("Price");
//Add rows to the DataTable

```

```

dataTable.Rows.Add(new object[] { "CA-1098", "Queso Cabrales", "12", "14",
"1", "167" });
dataTable.Rows.Add(new object[] { "LJ-0192-M", "Singaporean Hokkien Fried
Mee", "10", "20", "3", "197" });
dataTable.Rows.Add(new object[] { "SO-B909-M", "Mozzarella di Giovanni",
"15", "65", "10", "956" });
//Assign data source
pdfGrid.DataSource = dataTable;
//Draw grid to the page of PDF document
pdfGrid.Draw(page, new PointF(10, 10));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page
PdfPage page = doc.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ProductID");
dataTable.Columns.Add("ProductName");
dataTable.Columns.Add("Quantity");
dataTable.Columns.Add("UnitPrice");
dataTable.Columns.Add("Discount");
dataTable.Columns.Add("Price");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "CA-1098", "Queso Cabrales", "12", "14",
"1", "167" });
dataTable.Rows.Add(new object[] { "LJ-0192-M", "Singaporean Hokkien Fried
Mee", "10", "20", "3", "197" });
dataTable.Rows.Add(new object[] { "SO-B909-M", "Mozzarella di Giovanni",
"15", "65", "10", "956" });
//Assign data source
pdfGrid.DataSource = dataTable;
//Draw grid to the page of PDF document
pdfGrid.Draw(page, new PointF(10, 10));
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
doc.Save(stream);

```

```
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Creating a simple PDF document with basic elements

The [PdfDocument](#) object represents an entire PDF document that is being created. The following code example shows how to create a PDF document and add a [PdfPage](#) to it along with the [PdfPageSettings](#).

C#

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds page settings
document.PageSettings.Orientation = PdfPageOrientation.Landscape;
document.PageSettings.Margins.All = 50;
//Adds a page to the document
PdfPage page = document.Pages.Add();
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Adds page settings
document.PageSettings.Orientation = PdfPageOrientation.Landscape
document.PageSettings.Margins.All = 50
'Adds a page to the document
Dim page As PdfPage = document.Pages.Add()
```

UWP

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds page settings
document.PageSettings.Orientation = PdfPageOrientation.Landscape;
document.PageSettings.Margins.All = 50;
//Adds a page to the document
PdfPage page = document.Pages.Add();
```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds page settings
document.PageSettings.Orientation = PdfPageOrientation.Landscape;
document.PageSettings.Margins.All = 50;
//Adds a page to the document
PdfPage page = document.Pages.Add();
```

XAMARIN

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds page settings
document.PageSettings.Orientation = PdfPageOrientation.Landscape;
document.PageSettings.Margins.All = 50;
//Adds a page to the document
PdfPage page = document.Pages.Add();
```

1. Essential PDF has APIs similar to the .NET GDI plus which helps to draw elements to the PDF page just like 2D drawing in .NET.
2. Unlike System.Drawing APIs all the units are measured in point instead of pixel.
3. In PDF, all the elements are placed in absolute positions and has the possibility for content overlapping if misplaced.
4. Essential PDF provides the rendered bounds for each and every elements added, through [PdfLayoutResult](#) objects. This can be used to add successive elements and prevent content overlap.

The following code example explains how to add an image from disk to a PDF document, by providing the rectangle coordinates.

C#

```
//Loads the image from disk
PdfImage image = PdfImage.FromFile("AdventureCycle.jpg");
//Draws the image to the PDF page
page.Graphics.DrawImage(image, new RectangleF(176, 0, 390, 130));
```

VB.NET

```
'Loads the image from disk
Dim image As PdfImage = PdfImage.FromFile("AdventureCycle.jpg")
'Draws the image to the PDF page
page.Graphics.DrawImage(image, New RectangleF(176, 0, 390, 130))
```

UWP

```
//Loads the image as stream
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.Data.AdventureCycle.jpg");
PdfImage image = PdfImage.FromStream(imageStream);
//Draws the image to the PDF page
```

```
page.Graphics.DrawImage(image, new RectangleF(176, 0, 390, 130));
```

ASP.NET CORE

```
//Loads the image as stream
FileStream imageStream = new FileStream("AdventureCycle.jpg", FileMode.Open,
FileAccess.Read);
PdfImage image = PdfImage.FromStream(imageStream);
//Draws the image to the PDF page
page.Graphics.DrawImage(image, new RectangleF(176, 0, 390, 130));
```

XAMARIN

```
//Loads the image as stream
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
AdventureCycle.jpg");
PdfImage image = PdfImage.FromStream(imageStream);
//Draws the image to the PDF page
page.Graphics.DrawImage(image, new RectangleF(176, 0, 390, 130));
```

The following methods can be used to add text to a PDF document:

1. [DrawString\(\)](#) method of the [PdfGraphics](#)
2. [PdfTextElement](#) class.

The [PdfTextElement](#) provides the layout result of the added text by using the location of the next element that decides to prevent content overlapping. This is not available in the [DrawString](#) method.

The following code example adds the necessary text such as address, invoice number and date to create a basic invoice application.

C#

```
PdfLayoutResult result = new PdfLayoutResult(page, new RectangleF(0, 0,
page.Graphics.ClientSize.Width / 2, 95));
PdfFont subHeadingFont = new PdfStandardFont(PdfFontFamily.TimesRoman, 14);
//Draw Rectangle place on location
g.DrawRectangle(new PdfSolidBrush(new PdfColor(126, 151, 173)), new
RectangleF(0, result.Bounds.Bottom + 40, g.ClientSize.Width, 30));
element = new PdfTextElement("INVOICE " + 10248, subHeadingFont);
element.Brush = PdfBrushes.White;
result = element.Draw(page, new PointF(10, result.Bounds.Bottom + 48));
string currentDate = "DATE " + DateTime.Now.ToString("MM/dd/yyyy");
SizeF textSize = subHeadingFont.MeasureString(currentDate);
g.DrawString(currentDate, subHeadingFont, element.Brush, new
PointF(g.ClientSize.Width - textSize.Width - 10, result.Bounds.Y));
//Draw Bill header
element = new PdfTextElement("BILL TO ", new
PdfStandardFont(PdfFontFamily.TimesRoman, 10));
element.Brush = new PdfSolidBrush(new PdfColor(126, 155, 203));
result = element.Draw(page, new PointF(10, result.Bounds.Bottom + 25));
//Draw Bill address
```

```

element = new PdfTextElement(string.Format("{0}, {1}, {2}", "Vin et alcohol
Chevalier", "\n59 rue deb l'Abbaye ", " Reims, France"), new
PdfStandardFont(PdfFontFamily.TimesRoman, 10));
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
result = element.Draw(page, new RectangleF(10, result.Bounds.Bottom + 3,
g.ClientSize.Width / 2, 100));
//Draw Bill line
g.DrawLine(new PdfPen(new PdfColor(126, 151, 173), 0.70f), new PointF(0,
result.Bounds.Bottom + 3), new PointF(g.ClientSize.Width,
result.Bounds.Bottom + 3));

```

VB.NET

```

Dim subHeadingFont As PdfFont = New
PdfStandardFont(PdfFontFamily.TimesRoman, 14)
'Draw Rectangle place on location
g.DrawRectangle(New PdfSolidBrush(New PdfColor(126, 151, 173)), New
RectangleF(0, (result.Bounds.Bottom + 40), g.ClientSize.Width, 30))
element = New PdfTextElement(("INVOICE " + 10248), subHeadingFont)
element.Brush = PdfBrushes.White
result = element.Draw(page, New PointF(10, (result.Bounds.Bottom + 48)))
Dim currentDate As String = ("DATE " + DateTime.Now.ToString("MM/dd/yyyy"))
Dim textSize As SizeF = subHeadingFont.MeasureString(currentDate)
g.DrawString(currentDate, subHeadingFont, element.Brush, New
PointF((g.ClientSize.Width - (textSize.Width - 10)), result.Bounds.Y))
'Draw Bill header
element = New PdfTextElement("BILL TO ", New
PdfStandardFont(PdfFontFamily.TimesRoman, 10))
element.Brush = New PdfSolidBrush(New PdfColor(126, 155, 203))
result = element.Draw(page, New PointF(10, (result.Bounds.Bottom + 25)))
'Draw Bill address
element = New PdfTextElement(String.Format("{0}, {1}, {2}", "Vin et alcohol
Chevalier", "" & vbLf & "59 rue deb l'Abbaye ", " Reims, France"), New
PdfStandardFont(PdfFontFamily.TimesRoman, 10))
element.Brush = New PdfSolidBrush(New PdfColor(89, 89, 93))
result = element.Draw(page, New RectangleF(10, (result.Bounds.Bottom + 3),
(g.ClientSize.Width / 2), 100))
'Draw Bill line
g.DrawLine(New PdfPen(New PdfColor(126, 151, 173), 0.7!), New PointF(0,
(result.Bounds.Bottom + 3)), New PointF(g.ClientSize.Width,
(result.Bounds.Bottom + 3)))

```

UWP

```

PdfLayoutResult result = new PdfLayoutResult(page, new RectangleF(0, 0,
page.Graphics.ClientSize.Width / 2, 95));
PdfFont subHeadingFont = new PdfStandardFont(PdfFontFamily.TimesRoman, 14);
//Draw Rectangle place on location
g.DrawRectangle(new PdfSolidBrush(new PdfColor(126, 151, 173)), new
RectangleF(0, result.Bounds.Bottom + 40, g.ClientSize.Width, 30));
element = new PdfTextElement("INVOICE " + 10248, subHeadingFont);
element.Brush = PdfBrushes.White;
result = element.Draw(page, new PointF(10, result.Bounds.Bottom + 48));
string currentDate = "DATE " + DateTime.Now.ToString("MM/dd/yyyy");
SizeF textSize = subHeadingFont.MeasureString(currentDate);

```

```

g.DrawString(currentDate, subHeadingFont, element.Brush, new
PointF(g.ClientSize.Width - textSize.Width - 10, result.Bounds.Y));
//Draw Bill header
element = new PdfTextElement("BILL TO ", new
PdfStandardFont(PdfFontFamily.TimesRoman, 10));
element.Brush = new PdfSolidBrush(new PdfColor(126, 155, 203));
result = element.Draw(page, new PointF(10, result.Bounds.Bottom + 25));
//Draw Bill address
element = new PdfTextElement(string.Format("{0}, {1}, {2}", "Vin et alcohol
Chevalier", "\n59 rue deb l'Abbaye ", " Reims, France"), new
PdfStandardFont(PdfFontFamily.TimesRoman, 10));
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
result = element.Draw(page, new RectangleF(10, result.Bounds.Bottom + 3,
g.ClientSize.Width / 2, 100));
//Draw Bill line
g.DrawLine(new PdfPen(new PdfColor(126, 151, 173), 0.70f), new PointF(0,
result.Bounds.Bottom + 3), new PointF(g.ClientSize.Width,
result.Bounds.Bottom + 3));

```

ASP.NET CORE

```

PdfLayoutResult result = new PdfLayoutResult(page, new RectangleF(0, 0,
page.Graphics.ClientSize.Width / 2, 95));
PdfFont subHeadingFont = new PdfStandardFont(PdfFontFamily.TimesRoman, 14);
//Draw Rectangle place on location
g.DrawRectangle(new PdfSolidBrush(new PdfColor(126, 151, 173)), new
RectangleF(0, result.Bounds.Bottom + 40, g.ClientSize.Width, 30));
element = new PdfTextElement("INVOICE " + 10248, subHeadingFont);
element.Brush = PdfBrushes.White;
result = element.Draw(page, new PointF(10, result.Bounds.Bottom + 48));
string currentDate = "DATE " + DateTime.Now.ToString("MM/dd/yyyy");
SizeF textSize = subHeadingFont.MeasureString(currentDate);
g.DrawString(currentDate, subHeadingFont, element.Brush, new
PointF(g.ClientSize.Width - textSize.Width - 10, result.Bounds.Y));
//Draw Bill header
element = new PdfTextElement("BILL TO ", new
PdfStandardFont(PdfFontFamily.TimesRoman, 10));
element.Brush = new PdfSolidBrush(new PdfColor(126, 155, 203));
result = element.Draw(page, new PointF(10, result.Bounds.Bottom + 25));
//Draw Bill address
element = new PdfTextElement(string.Format("{0}, {1}, {2}", "Vin et alcohol
Chevalier", "\n59 rue deb l'Abbaye ", " Reims, France"), new
PdfStandardFont(PdfFontFamily.TimesRoman, 10));
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
result = element.Draw(page, new RectangleF(10, result.Bounds.Bottom + 3,
g.ClientSize.Width / 2, 100));
//Draw Bill line
g.DrawLine(new PdfPen(new PdfColor(126, 151, 173), 0.70f), new PointF(0,
result.Bounds.Bottom + 3), new PointF(g.ClientSize.Width,
result.Bounds.Bottom + 3));

```

XAMARIN

```

PdfLayoutResult result = new PdfLayoutResult(page, new RectangleF(0, 0,
page.Graphics.ClientSize.Width / 2, 95));
PdfFont subHeadingFont = new PdfStandardFont(PdfFontFamily.TimesRoman, 14);

```

```
//Draw Rectangle place on location
g.DrawRectangle(new PdfSolidBrush(new PdfColor(126, 151, 173)), new
RectangleF(0, result.Bounds.Bottom + 40, g.ClientSize.Width, 30));
element = new PdfTextElement("INVOICE " + 10248, subHeadingFont);
element.Brush = PdfBrushes.White;
result = element.Draw(page, new PointF(10, result.Bounds.Bottom + 48));
string currentDate = "DATE " + DateTime.Now.ToString("MM/dd/yyyy");
SizeF textSize = subHeadingFont.MeasureString(currentDate);
g.DrawString(currentDate, subHeadingFont, element.Brush, new
PointF(g.ClientSize.Width - textSize.Width - 10, result.Bounds.Y));
//Draw Bill header
element = new PdfTextElement("BILL TO ", new
PdfStandardFont(PdfFontFamily.TimesRoman, 10));
element.Brush = new PdfSolidBrush(new PdfColor(126, 155, 203));
result = element.Draw(page, new PointF(10, result.Bounds.Bottom + 25));
//Draw Bill address
element = new PdfTextElement(string.Format("{0}, {1}, {2}", "Vin et alcohol
Chevalier", "\n59 rue deb l'Abbaye ", " Reims, France"), new
PdfStandardFont(PdfFontFamily.TimesRoman, 10));
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
result = element.Draw(page, new RectangleF(10, result.Bounds.Bottom + 3,
g.ClientSize.Width / 2, 100));
//Draw Bill line
g.DrawLine(new PdfPen(new PdfColor(126, 151, 173), 0.70f), new PointF(0,
result.Bounds.Bottom + 3), new PointF(g.ClientSize.Width,
result.Bounds.Bottom + 3));
```

Essential PDF provides two types of table models. The difference between both the table models can be referred from the link

[Difference between PdfLightTable and PdfGrid](#)

Since the invoice document requires only simple cell customizations, the given code example explains how to create a simple invoice table by using [PdfGrid](#).

C#

```
//Creates the datasource for the table
DataTable invoiceDetails = GetProductDetailsAsDataTable();
//Creates a PDF grid
PdfGrid grid = new PdfGrid();
//Adds the data source
grid.DataSource = invoiceDetails;
//Creates the grid cell styles
PdfGridCellStyle cellStyle = new PdfGridCellStyle();
cellStyle.Borders.All = PdfPens.White;
PdfGridRow header = grid.Headers[0];
//Creates the header style
PdfGridCellStyle headerStyle = new PdfGridCellStyle();
headerStyle.Borders.All = new PdfPen(new PdfColor(126, 151, 173));
headerStyle.BackgroundBrush = new PdfSolidBrush(new PdfColor(126, 151,
173));
headerStyle.TextBrush = PdfBrushes.White;
headerStyle.Font = new PdfStandardFont(PdfFontFamily.TimesRoman, 14f,
PdfFontStyle.Regular);
//Adds cell customizations
for (int i = 0; i < header.Cells.Count; i++)
```



```

{
    if (i == 0 || i == 1)
        header.Cells[i].StringFormat = new PdfStringFormat(PdfTextAlignment.Left,
            PdfVerticalAlignment.Middle);
    else
        header.Cells[i].StringFormat = new PdfStringFormat(PdfTextAlignment.Right,
            PdfVerticalAlignment.Middle);
}
//Applies the header style
header.ApplyStyle(headerStyle);
cellStyle.Borders.Bottom = new PdfPen(new PdfColor(217, 217, 217), 0.70f);
cellStyle.Font = new PdfStandardFont(PdfFontFamily.TimesRoman, 12f);
cellStyle.TextBrush = new PdfSolidBrush(new PdfColor(131, 130, 136));
//Creates the layout format for grid
PdfGridLayoutFormat layoutFormat = new PdfGridLayoutFormat();
// Creates layout format settings to allow the table pagination
layoutFormat.Layout = PdfLayoutType.Paginate;
//Draws the grid to the PDF page.
PdfGridLayoutResult gridResult = grid.Draw(page, new RectangleF(new
    PointF(0, result.Bounds.Bottom + 40), new SizeF(g.ClientSize.Width,
        g.ClientSize.Height - 100)), layoutFormat);

```

VB.NET

```

'Creates the datasource for the table
Dim invoiceDetails As DataTable =
    GetProductDetails(Integer.Parse(invoiceNumber))
'Create a PDF grid
Dim grid As New PdfGrid()
'Adds the data source
grid.DataSource = invoiceDetails
'creates the grid cell styles
Dim cellStyle As New PdfGridCellStyle()
cellStyle.Borders.All = PdfPens.White
Dim header As PdfGridRow = grid.Headers(0)
'Creates the header style
Dim headerStyle As New PdfGridCellStyle()
headerStyle.Borders.All = New PdfPen(New PdfColor(126, 151, 173))
headerStyle.BackgroundBrush = New PdfSolidBrush(New PdfColor(126, 151, 173))
headerStyle.TextBrush = PdfBrushes.White
headerStyle.Font = New PdfStandardFont(PdfFontFamily.TimesRoman, 14.0F,
    PdfFontStyle.Regular)
'Adds cell customizations
For i As Integer = 0 To header.Cells.Count - 1
    If i = 0 OrElse i = 1 Then
        header.Cells(i).StringFormat = New PdfStringFormat(PdfTextAlignment.Left,
            PdfVerticalAlignment.Middle)
    Else
        header.Cells(i).StringFormat = New PdfStringFormat(PdfTextAlignment.Right,
            PdfVerticalAlignment.Middle)
    End If
Next
'Applies the header style
header.ApplyStyle(headerStyle)
cellStyle.Borders.Bottom = New PdfPen(New PdfColor(217, 217, 217), 0.7F)
cellStyle.Font = New PdfStandardFont(PdfFontFamily.TimesRoman, 12.0F)

```

```

cellStyle.TextBrush = New PdfSolidBrush(New PdfColor(131, 130, 136))
'Creates the layout format for grid
Dim layoutFormat As New PdfGridLayoutFormat()
'Layout format settings to allow the table pagination
layoutFormat.Layout = PdfLayoutType.Paginate
'Draws the grid to the PDF page.
Dim gridResult As PdfGridLayoutResult = grid.Draw(page, New RectangleF(New
PointF(0, result.Bounds.Bottom + 40), New SizeF(g.ClientSize.Width,
g.ClientSize.Height - 100)), layoutFormat)

```

UWP

```

//Creates the datasource for the table
DataTable invoiceDetails = GetProductDetailsAsDataTable();
//Creates a PDF grid
PdfGrid grid = new PdfGrid();
//Adds the data source
grid.DataSource = invoiceDetails;
//Creates the grid cell styles
PdfGridCellStyle cellStyle = new PdfGridCellStyle();
cellStyle.Borders.All = PdfPens.White;
PdfGridRow header = grid.Headers[0];
//Creates the header style
PdfGridCellStyle headerStyle = new PdfGridCellStyle();
headerStyle.Borders.All = new PdfPen(new PdfColor(126, 151, 173));
headerStyle.BackgroundBrush = new PdfSolidBrush(new PdfColor(126, 151,
173));
headerStyle.TextBrush = PdfBrushes.White;
headerStyle.Font = new PdfStandardFont(PdfFontFamily.TimesRoman, 14f,
PdfFontStyle.Regular);
//Adds cell customizations
for (int i = 0; i < header.Cells.Count; i++)
{
    if (i == 0 || i == 1)
        header.Cells[i].StringFormat = new PdfStringFormat(PdfTextAlignment.Left,
PdfVerticalAlignment.Middle);
    else
        header.Cells[i].StringFormat = new PdfStringFormat(PdfTextAlignment.Right,
PdfVerticalAlignment.Middle);
}
//Applies the header style
header.ApplyStyle(headerStyle);
cellStyle.Borders.Bottom = new PdfPen(new PdfColor(217, 217, 217), 0.70f);
cellStyle.Font = new PdfStandardFont(PdfFontFamily.TimesRoman, 12f);
cellStyle.TextBrush = new PdfSolidBrush(new PdfColor(131, 130, 136));
foreach (PdfGridRow row in grid.Rows)
{
    row.ApplyStyle(cellStyle);
    for (int i = 0; i < row.Cells.Count; i++)
    {
        PdfGridCell cell = row.Cells[i];
        if (i == 1)
            cell.StringFormat = new PdfStringFormat(PdfTextAlignment.Left,
PdfVerticalAlignment.Middle);
        else if (i == 0)

```

```

cell.StringFormat = new PdfStringFormat(PdfTextAlignment.Center,
PdfVerticalAlignment.Middle);
else
cell.StringFormat = new PdfStringFormat(PdfTextAlignment.Right,
PdfVerticalAlignment.Middle);
if (i > 2)
{
float val = float.MinValue;
float.TryParse(cell.Value.ToString(), out val);
cell.Value = val.ToString("C");
}
}
}
grid.Columns[0].Width = 100;
grid.Columns[1].Width = 200;
//Creates the layout format for grid
PdfGridLayoutFormat layoutFormat = new PdfGridLayoutFormat();
//Creates layout format settings to allow the table pagination
layoutFormat.Layout = PdfLayoutType.Paginate;
//Draws the grid to the PDF page
PdfGridLayoutResult gridResult = grid.Draw(page, new RectangleF(new
PointF(0, result.Bounds.Bottom + 40), new SizeF(g.ClientSize.Width,
g.ClientSize.Height - 100)), layoutFormat);
float pos = 0.0f;
for (int i = 0; i < grid.Columns.Count - 1; i++)
pos += grid.Columns[i].Width;
PdfFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 14f);
gridResult.Page.Graphics.DrawString("Total Due", font, new PdfSolidBrush(new
PdfColor(126, 151, 173)), new RectangleF(new PointF(pos,
gridResult.Bounds.Bottom + 20), new SizeF(grid.Columns[3].Width - pos, 20)),
new PdfStringFormat(PdfTextAlignment.Right));
gridResult.Page.Graphics.DrawString("Thank you for your business!", new
PdfStandardFont(PdfFontFamily.TimesRoman, 12), new PdfSolidBrush(new
PdfColor(89, 89, 93)), new PointF(pos - 55, gridResult.Bounds.Bottom + 60));
pos += grid.Columns[4].Width;
gridResult.Page.Graphics.DrawString(total.ToString("C"), font, new
PdfSolidBrush(new PdfColor(131, 130, 136)), new RectangleF(new PointF(pos,
gridResult.Bounds.Bottom + 20), new SizeF(grid.Columns[4].Width - pos, 20)),
new PdfStringFormat(PdfTextAlignment.Right));

```

ASP.NET CORE

```

//Creates the datasource for the table
DataTable invoiceDetails = GetProductDetailsAsDataTable();
//Creates a PDF grid
PdfGrid grid = new PdfGrid();
//Adds the data source
grid.DataSource = invoiceDetails;
//Creates the grid cell styles
PdfGridCellStyle cellStyle = new PdfGridCellStyle();
cellStyle.Borders.All = PdfPens.White;
PdfGridRow header = grid.Headers[0];
//Creates the header style
PdfGridCellStyle headerStyle = new PdfGridCellStyle();
headerStyle.Borders.All = new PdfPen(new PdfColor(126, 151, 173));

```

```

headerStyle.BackgroundBrush = new PdfSolidBrush(new PdfColor(126, 151,
173));
headerStyle.TextBrush = PdfBrushes.White;
headerStyle.Font = new PdfStandardFont(PdfFontFamily.TimesRoman, 14f,
PdfFontStyle.Regular);
//Adds cell customizations
for (int i = 0; i < header.Cells.Count; i++)
{
    if (i == 0 || i == 1)
        header.Cells[i].StringFormat = new PdfStringFormat(PdfTextAlignment.Left,
PdfVerticalAlignment.Middle);
    else
        header.Cells[i].StringFormat = new PdfStringFormat(PdfTextAlignment.Right,
PdfVerticalAlignment.Middle);
}
//Applies the header style
header.ApplyStyle(headerStyle);
cellStyle.Borders.Bottom = new PdfPen(new PdfColor(217, 217, 217), 0.70f);
cellStyle.Font = new PdfStandardFont(PdfFontFamily.TimesRoman, 12f);
cellStyle.TextBrush = new PdfSolidBrush(new PdfColor(131, 130, 136));
foreach (PdfGridRow row in grid.Rows)
{
    row.ApplyStyle(cellStyle);
    for (int i = 0; i < row.Cells.Count; i++)
    {
        PdfGridCell cell = row.Cells[i];
        if (i == 1)
            cell.StringFormat = new PdfStringFormat(PdfTextAlignment.Left,
PdfVerticalAlignment.Middle);
        else if (i == 0)
            cell.StringFormat = new PdfStringFormat(PdfTextAlignment.Center,
PdfVerticalAlignment.Middle);
        else
            cell.StringFormat = new PdfStringFormat(PdfTextAlignment.Right,
PdfVerticalAlignment.Middle);
        if (i > 2)
        {
            float val = float.MinValue;
            float.TryParse(cell.Value.ToString(), out val);
            cell.Value = val.ToString("C");
        }
    }
}
grid.Columns[0].Width = 100;
grid.Columns[1].Width = 200;
//Creates the layout format for grid
PdfGridLayoutFormat layoutFormat = new PdfGridLayoutFormat();
//Creates layout format settings to allow the table pagination
layoutFormat.Layout = PdfLayoutType.Paginate;
//Draws the grid to the PDF page
PdfGridLayoutResult gridResult = grid.Draw(page, new RectangleF(new
PointF(0, result.Bounds.Bottom + 40), new SizeF(g.ClientSize.Width,
g.ClientSize.Height - 100)), layoutFormat);
float pos = 0.0f;
for (int i = 0; i < grid.Columns.Count - 1; i++)
    pos += grid.Columns[i].Width;
PdfFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 14f);

```

```

gridResult.Page.Graphics.DrawString("Total Due", font, new PdfSolidBrush(new
PdfColor(126, 151, 173)), new RectangleF(new PointF(pos,
gridResult.Bounds.Bottom + 20), new SizeF(grid.Columns[3].Width - pos, 20)),
new PdfStringFormat(PdfTextAlignment.Right));
gridResult.Page.Graphics.DrawString("Thank you for your business!", new
PdfStandardFont(PdfFontFamily.TimesRoman, 12), new PdfSolidBrush(new
PdfColor(89, 89, 93)), new PointF(pos - 55, gridResult.Bounds.Bottom + 60));
pos += grid.Columns[4].Width;
gridResult.Page.Graphics.DrawString(total.ToString("C"), font, new
PdfSolidBrush(new PdfColor(131, 130, 136)), new RectangleF(new PointF(pos,
gridResult.Bounds.Bottom + 20), new SizeF(grid.Columns[4].Width - pos, 20)),
new PdfStringFormat(PdfTextAlignment.Right));

```

XAMARIN

```

//Creates the datasource for the table
DataTable invoiceDetails = GetProductDetailsAsDataTable();
//Creates a PDF grid
PdfGrid grid = new PdfGrid();
//Adds the data source
grid.DataSource = invoiceDetails;
//Creates the grid cell styles
PdfGridCellStyle cellStyle = new PdfGridCellStyle();
cellStyle.Borders.All = PdfPens.White;
PdfGridRow header = grid.Headers[0];
//Creates the header style
PdfGridCellStyle headerStyle = new PdfGridCellStyle();
headerStyle.Borders.All = new PdfPen(new PdfColor(126, 151, 173));
headerStyle.BackgroundBrush = new PdfSolidBrush(new PdfColor(126, 151,
173));
headerStyle.TextBrush = PdfBrushes.White;
headerStyle.Font = new PdfStandardFont(PdfFontFamily.TimesRoman, 14f,
PdfFontStyle.Regular);
//Adds cell customizations
for (int i = 0; i < header.Cells.Count; i++)
{
    if (i == 0 || i == 1)
        header.Cells[i].StringFormat = new PdfStringFormat(PdfTextAlignment.Left,
        PdfVerticalAlignment.Middle);
    else
        header.Cells[i].StringFormat = new PdfStringFormat(PdfTextAlignment.Right,
        PdfVerticalAlignment.Middle);
}
//Applies the header style
header.ApplyStyle(headerStyle);
cellStyle.Borders.Bottom = new PdfPen(new PdfColor(217, 217, 217), 0.70f);
cellStyle.Font = new PdfStandardFont(PdfFontFamily.TimesRoman, 12f);
cellStyle.TextBrush = new PdfSolidBrush(new PdfColor(131, 130, 136));
foreach (PdfGridRow row in grid.Rows)
{
    row.ApplyStyle(cellStyle);
    for (int i = 0; i < row.Cells.Count; i++)
    {
        PdfGridCell cell = row.Cells[i];
        if (i == 1)

```

```

cell.StringFormat = new PdfStringFormat(PdfTextAlignment.Left,
PdfVerticalAlignment.Middle);
else if (i == 0)
cell.StringFormat = new PdfStringFormat(PdfTextAlignment.Center,
PdfVerticalAlignment.Middle);
else
cell.StringFormat = new PdfStringFormat(PdfTextAlignment.Right,
PdfVerticalAlignment.Middle);
if (i > 2)
{
float val = float.MinValue;
float.TryParse(cell.Value.ToString(), out val);
cell.Value = val.ToString("C");
}
}
}
grid.Columns[0].Width = 100;
grid.Columns[1].Width = 200;
//Creates the layout format for grid
PdfGridLayoutFormat layoutFormat = new PdfGridLayoutFormat();
//Creates layout format settings to allow the table pagination
layoutFormat.Layout = PdfLayoutType.Paginate;
//Draws the grid to the PDF page
PdfGridLayoutResult gridResult = grid.Draw(page, new RectangleF(new
PointF(0, result.Bounds.Bottom + 40), new SizeF(g.ClientSize.Width,
g.ClientSize.Height - 100)), layoutFormat);
float pos = 0.0f;
for (int i = 0; i < grid.Columns.Count - 1; i++)
pos += grid.Columns[i].Width;
PdfFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 14f);
gridResult.Page.Graphics.DrawString("Total Due", font, new PdfSolidBrush(new
PdfColor(126, 151, 173)), new RectangleF(new PointF(pos,
gridResult.Bounds.Bottom + 20), new SizeF(grid.Columns[3].Width - pos, 20)),
new PdfStringFormat(PdfTextAlignment.Right));
gridResult.Page.Graphics.DrawString("Thank you for your business!", new
PdfStandardFont(PdfFontFamily.TimesRoman, 12), new PdfSolidBrush(new
PdfColor(89, 89, 93)), new PointF(pos - 55, gridResult.Bounds.Bottom + 60));
pos += grid.Columns[4].Width;
gridResult.Page.Graphics.DrawString(total.ToString("C"), font, new
PdfSolidBrush(new PdfColor(131, 130, 136)), new RectangleF(new PointF(pos,
gridResult.Bounds.Bottom + 20), new SizeF(grid.Columns[4].Width - pos, 20)),
new PdfStringFormat(PdfTextAlignment.Right));

```

The following code example shows how to save the invoice document to disk and dispose the [PdfDocument](#) object.

C#

```

//Saves and closes the document.
document.Save("Sample.pdf");
document.Close(true);

```

VB.NET

```

'Saves and closes the document.
document.Save("Sample.pdf")

```

```
document.Close(True)
```

UWP

```
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Sample.pdf");
```

ASP.NET CORE

```
FileStream fileStream = new FileStream("Sample.pdf", FileMode.CreateNew,
FileAccess.ReadWrite);
//Save and close the PDF document
document.Save(fileStream);
document.Close(true);
```

XAMARIN

```
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pdf",
"application/pdf", stream);
}
```

The following screenshot shows the invoice PDF document created by using Essential PDF.

ADVENTURE WORKS cycles

INVOICE 10248 **DATE 08/08/2015**

BILL TO
 Vin et alcools Chevalier
 59 rue de Talmaye, Reims, France

ProductID	ProductName	Quantity	UnitPrice	Discount	Price
CA-1095	Queso Cabrales	12	\$14	\$1	\$167
LJ-0192-M	Singaporean Hokkien Fried Mee	10	\$20	\$3	\$197
SO-B909-M	Mozzarella di Giovanni	15	\$65	\$10	\$965
Total Due					1329

Thank you for your business!

Filling forms

An interactive form, sometimes referred to as an AcroForm is a collection of fields for gathering information interactively from the user. A PDF document can contain any number of fields appearing in any combination of pages, all of that make a single, globally interactive form spanning the entire document.

Essential PDF allows you to create and manipulate existing form in PDF document. To work with existing form documents, the following namespaces are required.

1. Syncfusion.Pdf
2. Syncfusion.Pdf.Parsing

The following guide shows how to fill a sample PDF form as shown.

The screenshot shows a PDF form titled "JOB APPLICATION FORM" open in Adobe Reader. The form has the following fields:

- First name:** A single-line text input field.
- Last Name:** A single-line text input field.
- Address:** A multi-line text input field.
- Gender:** Two radio button options labeled "Male" and "Female".
- Occupation:** Four checkbox options labeled "Student", "Administrative/Clerk", "Business", and "Retiree".

The Adobe Reader interface is visible at the top, showing the menu bar (File, Edit, View, Window, Help) and a toolbar with icons for opening, saving, printing, and commenting. A status bar at the bottom indicates the file is "JobApplication.pdf" and is at 115% zoom.

Essential PDF allows you to fill the form fields by using [PdfLoadedField](#) class. You can get the form field either by using its field name or field index.

C#

```
//Loads the PDF form.
PdfLoadedDocument loadedDocument = new
PdfLoadedDocument(@"JobApplication.pdf");
//Loads the form
PdfLoadedForm form = loadedDocument.Form;
//Fills the textbox field by using index
(form.Fields[0] as PdfLoadedTextBoxField).Text = "John";
//Fills the textbox fields by using field name
(form.Fields["LastName"] as PdfLoadedTextBoxField).Text = "Doe";
(form.Fields["Address"] as PdfLoadedTextBoxField).Text = " John Doe \n 123
Main St \n Anytown, USA";
//Loads the radio button group
PdfLoadedRadioButtonItemCollection radioButtonCollection =
(form.Fields["Gender"] as PdfLoadedRadioButtonListField).Items;
//Checks the 'Male' option
radioButtonCollection[0].Checked = true;
//Checks the 'business' checkbox field
(form.Fields["Business"] as PdfLoadedCheckBoxField).Checked = true;
//Checks the 'retiree' checkbox field
(form.Fields["Retiree"] as PdfLoadedCheckBoxField).Checked = true;
//Saves and closes the document
loadedDocument.Save("output.pdf");
loadedDocument.Close(true);
```

VB.NET

```

'Loads the PDF form.
Dim loadedDocument As New PdfLoadedDocument("JobApplication.pdf")
'Load the form
Dim form As PdfLoadedForm = loadedDocument.Form
'Fills the textbox field by using index
TryCast(form.Fields(0), PdfLoadedTextBoxField).Text = "John"
'Fills the textbox fields by using field name
TryCast(form.Fields("LastName"), PdfLoadedTextBoxField).Text = "Doe"
TryCast(form.Fields("Address"), PdfLoadedTextBoxField).Text = " John Doe " &
vbLf & " 123 Main St " & vbLf & " Anytown, USA"
'Load the radio button group
Dim radioButtonCollection As PdfLoadedRadioButtonItemCollection =
TryCast(form.Fields("Gender"), PdfLoadedRadioButtonListField).Items
'Checks the 'Male' option
radioButtonCollection(0).Checked = True
'Checks the 'business' checkbox field
TryCast(form.Fields("Business"), PdfLoadedCheckBoxField).Checked = True
'Checks the 'retiree' checkbox field
TryCast(form.Fields("Retiree"), PdfLoadedCheckBoxField).Checked = True
'Saves and closes the document
loadedDocument.Save("output.pdf")
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Loads the form
PdfLoadedForm form = loadedDocument.Form;
//Fills the textbox field by using index
(form.Fields[0] as PdfLoadedTextBoxField).Text = "John";
//Fills the textbox fields by using field name
(form.Fields["LastName"] as PdfLoadedTextBoxField).Text = "Doe";
(form.Fields["Address"] as PdfLoadedTextBoxField).Text = " John Doe \n 123
Main St \n Anytown, USA";
//Loads the radio button group
PdfLoadedRadioButtonItemCollection radioButtonCollection =
(form.Fields["Gender"] as PdfLoadedRadioButtonListField).Items;
//Checks the 'Male' option
radioButtonCollection[0].Checked = true;
//Checks the 'business' checkbox field
(form.Fields["Business"] as PdfLoadedCheckBoxField).Checked = true;
//Checks the 'retiree' checkbox field
(form.Fields["Retiree"] as PdfLoadedCheckBoxField).Checked = true;
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.

```

```
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("JobApplication.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Loads the form
PdfLoadedForm form = loadedDocument.Form;
//Fills the textbox field by using index
(form.Fields[0] as PdfLoadedTextBoxField).Text = "John";
//Fills the textbox fields by using field name
(form.Fields["LastName"] as PdfLoadedTextBoxField).Text = "Doe";
(form.Fields["Address"] as PdfLoadedTextBoxField).Text = " John Doe \n 123
Main St \n Anytown, USA";
//Loads the radio button group
PdfLoadedRadioButtonItemCollection radioButtonCollection =
(form.Fields["Gender"] as PdfLoadedRadioButtonListField).Items;
//Checks the 'Male' option
radioButtonCollection[0].Checked = true;
//Checks the 'business' checkbox field
(form.Fields["Business"] as PdfLoadedCheckBoxField).Checked = true;
//Checks the 'retiree' checkbox field
(form.Fields["Retiree"] as PdfLoadedCheckBoxField).Checked = true;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
JobApplication.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Loads the form
PdfLoadedForm form = loadedDocument.Form;
//Fills the textbox field by using index
(form.Fields[0] as PdfLoadedTextBoxField).Text = "John";
//Fills the textbox fields by using field name
(form.Fields["LastName"] as PdfLoadedTextBoxField).Text = "Doe";
```

```

(form.Fields["Address"] as PdfLoadedTextBoxField).Text = " John Doe \n 123
Main St \n Anytown, USA";
//Loads the radio button group
PdfLoadedRadioButtonItemCollection radioButtonCollection =
(form.Fields["Gender"] as PdfLoadedRadioButtonListField).Items;
//Checks the 'Male' option
radioButtonCollection[0].Checked = true;
//Checks the 'business' checkbox field
(form.Fields["Business"] as PdfLoadedCheckBoxField).Checked = true;
//Checks the 'retiree' checkbox field
(form.Fields["Retiree"] as PdfLoadedCheckBoxField).Checked = true;
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
"application/pdf", stream);
}

```

The filled form is shown in adobe reader application as follows.

The screenshot shows a PDF document titled 'filledform.pdf' in Adobe Reader. The form is titled 'JOB APPLICATION FORM'. It has the following fields and values:

- First name: John
- Last Name: Doe
- Address: John Doe, 123 Main St, Anytown, USA
- Gender: Male (selected with a radio button)
- Occupation: Student (unchecked), Administrative/Clerk (unchecked), Business (checked), Retiree (checked)

Converting HTML contents to PDF

Essential PDF supports converting HTML contents to PDF. To add the HTML to PDF conversion functionality by using WebKit rendering engine, the following assemblies need to be added as reference to the project.

Assembly Name	Description
Syncfusion.Pdf.Base	This assembly contains the core feature for creating, manipulating and saving PDF documents.
Syncfusion.Compression.Base	This assembly is required for compressing the internal contents of a PDF document.
Syncfusion.HtmlConverter.Base.dll	This assembly is required for converting the HTML to PDF

The QtBinaries available in the WebKitHTMLConverter installed location (**\$System drive\Program Files(x86)\Syncfusion\WebKitHTMLConverter\xx.x.x.xx\QtBinaries**) should be placed in the local machine where the conversion takes place. The physical path of this folder has been set to the [WebKitPath](#) property of the [WebKitConverterSettings](#) class, as shown. By default it will search for WebKit assemblies in bin folder.

C#

```
//Create a WebKitConverterSettings instance
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"QtBinaries/";
```

VB.NET

```
'Create a WebKitConverterSettings instance
Dim webKitSettings As New WebKitConverterSettings()
'Set WebKit path
webKitSettings.WebKitPath = "/QtBinaries/"
```

UWP

```
//PDF supports converting HTML contents to PDF only in Windows Forms, WPF,
ASP.NET, ASP.NET MVC and ASP.NET Core platforms.
```

ASP.NET CORE

```
//Create a WebKitConverterSettings instance
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"/QtBinaries/";
```

XAMARIN

```
//PDF supports converting HTML contents to PDF only in Windows Forms, WPF,
ASP.NET, ASP.NET MVC and ASP.NET Core platforms.
```

For converting HTTPS sites, it requires OPENSSL libraries to be installed in the machine. You can install the OPENSSL library by downloading its setup from the following link,

[OpenSSL](#)

WebKit conversion also requires VC++ 2010 redistributable to be installed in the machine. You can use the below mentioned download link,

X86 - <https://www.microsoft.com/en-in/download/details.aspx?id=5555>

X64 - <https://www.microsoft.com/en-in/download/details.aspx?id=14632>

To convert website URL or local HTML file to PDF by using WebKit rendering engine, refer to the following code example.

C#

```
//Create an instance of HTML to PDF converter
HtmlToPdfConverter htmlConverter = new
HtmlToPdfConverter(HtmlRenderingEngine.WebKit);
//Create a WebKitConverterSettings instance
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"/QtBinaries/";
//Assign the WebKit settings to converter
htmlConverter.ConverterSettings = webKitSettings;
//Convert the URL to PDF
PdfDocument document = htmlConverter.Convert("http://www.google.com");
//Save and close the document.
document.Save("Sample.pdf");
document.Close(true);
```

VB.NET

```
'Create an instance of HTML to PDF converter
Dim htmlConverter As New HtmlToPdfConverter(HtmlRenderingEngine.WebKit)
'Create a WebKitConverterSettings instance
Dim webKitSettings As New WebKitConverterSettings()
'Set WebKit path
webKitSettings.WebKitPath = "/QtBinaries/"
'Assign the WebKit settings to converter
htmlConverter.ConverterSettings = webKitSettings
'Convert the URL to PDF
Dim document As PdfDocument = htmlConverter.Convert("http://www.google.com")
'Save and close the document.
document.Save("Sample.pdf")
document.Close(True)
```

UWP

```
//PDF supports converting HTML contents to PDF only in Windows Forms, WPF,
ASP.NET, ASP.NET MVC and ASP.NET Core platforms.
```

ASP.NET CORE

```
//Create an instance of HTML to PDF converter
HtmlToPdfConverter htmlConverter = new HtmlToPdfConverter();
//Create a WebKitConverterSettings instance
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"/QtBinaries/";
//Assign the WebKit settings to converter
htmlConverter.ConverterSettings = webKitSettings;
//Convert the URL to PDF
PdfDocument document = htmlConverter.Convert("http://www.google.com");
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//PDF supports converting HTML contents to PDF only in Windows Forms, WPF,
ASP.NET, ASP.NET MVC and ASP.NET Core platforms.
```

To convert the HTML string to PDF, use the following code example.

C#

```
//Create an instance of HTML to PDF converter
HtmlToPdfConverter htmlConverter = new
HtmlToPdfConverter(HtmlRenderingEngine.WebKit);
//Create a WebKitConverterSettings instance
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"QtBinaries/";
//Assign the WebKit settings to converter
htmlConverter.ConverterSettings = webKitSettings;
//Convert the HTML string to PDF
PdfDocument document =
htmlConverter.Convert("<html><head><title></title></head><body><div>Hello
World!!!</div></body></html>", "");
//Save and close the document.
document.Save("Sample.pdf");
document.Close(true);
```

VB.NET

```
'Create an instance of HTML to PDF converter
Dim htmlConverter As New HtmlToPdfConverter(HtmlRenderingEngine.WebKit)
'Create a WebKitConverterSettings instance
Dim webKitSettings As New WebKitConverterSettings()
'Set WebKit path
webKitSettings.WebKitPath = "QtBinaries/"
'Assign the WebKit settings to converter
htmlConverter.ConverterSettings = webKitSettings
'Convert the HTML string to PDF
Dim document As PdfDocument =
htmlConverter.Convert("<html><head><title></title></head><body><div>Hello
World!!!</div></body></html>", "")
'Save and close the document.
document.Save("Sample.pdf")
document.Close(True)
```

UWP

```
//PDF supports converting HTML contents to PDF only in Windows Forms, WPF,
ASP.NET, ASP.NET MVC and ASP.NET Core platforms.
```

ASP.NET CORE

```
//Create an instance of HTML to PDF converter
HtmlToPdfConverter htmlConverter = new HtmlToPdfConverter();
//Create a WebKitConverterSettings instance
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"QtBinaries/";
//Assign the WebKit settings to converter
htmlConverter.ConverterSettings = webKitSettings;
//Convert the HTML string to PDF
PdfDocument document =
htmlConverter.Convert("<html><head><title></title></head><body><div>Hello
World!!!</div></body></html>", "");
//Save the document into stream
```



```
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//PDF supports converting HTML contents to PDF only in Windows Forms, WPF,
ASP.NET, ASP.NET MVC and ASP.NET Core platforms.
```

Merge PDF Documents

Essential PDF supports merging multiple PDF documents from disk and stream using [Merge](#) method. You can merge the multiple PDF documents from disk by specifying the path of the documents in a string array.

Refer to the following code example to merge multiple documents from disk.

C#

```
//Creates the new PDF document
PdfDocument finalDoc = new PdfDocument();
// Creates a string array of source files to be merged.
string[] source = { "file1.pdf, file2.pdf" };
// Merges PDFDocument.
PdfDocument.Merge(finalDoc, source);
//Saves the final document
finalDoc.Save("Sample.pdf");
//closes the document
finalDoc.Close(true);
```

VB.NET

```
'Creates the new PDF document
Dim finalDoc As New PdfDocument()
' Creates a string array of source files to be merged.
Dim source As String() = {"file1.pdf, file2.pdf"}
' Merges PDFDocument.
PdfDocument.Merge(finalDoc, source)
'Saves the final document
finalDoc.Save("Sample.pdf")
'closes the document
finalDoc.Close(True)
```

UWP

```
//PDF supports merging multiple PDF documents from disk only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//PDF supports merging multiple PDF documents from disk only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

XAMARIN

```
//PDF supports merging multiple PDF documents from disk only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

You can merge the PDF document streams by using the following code example.

C#

```
//Creates the destination document
PdfDocument finalDoc = new PdfDocument();
Stream stream1 = File.OpenRead("file1.pdf");
Stream stream2 = File.OpenRead("file2.pdf");
// Creates a PDF stream for merging.
Stream[] streams = { stream1, stream2 };
// Merges PDFDocument.
PdfDocumentBase.Merge(finalDoc, streams);
//Saves the document
finalDoc.Save("sample.pdf");
//closes the document
finalDoc.Close(true);
```

VB.NET

```
'creates the destination document
Dim finalDoc As New PdfDocument()
Dim stream1 As Stream = File.OpenRead("file1.pdf")
Dim stream2 As Stream = File.OpenRead("file2.pdf")
' Creates a PDF stream for merging.
Dim streams As Stream() = {stream1, stream2}
' Merges PDFDocument.
PdfDocumentBase.Merge(finalDoc, streams)
'Saves the document
finalDoc.Save("sample.pdf")
'closes the document
finalDoc.Close(True)
```

UWP

```
//PDF supports merging multiple PDF documents from stream only in Windows Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Core platforms.
```

ASP.NET CORE

```
//Creates a PDF document
PdfDocument finalDoc = new PdfDocument();
```

```

FileStream stream1 = new FileStream("file1.pdf", FileMode.Open,
FileAccess.Read);
FileStream stream2 = new FileStream("file2.pdf", FileMode.Open,
FileAccess.Read);
// Creates a PDF stream for merging
Stream[] streams = { stream1, stream2 };
// Merges PDFDocument.
PdfDocumentBase.Merge(finalDoc, streams);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
finalDoc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
finalDoc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//PDF supports merging multiple PDF documents from stream only in Windows
Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Core platforms.

```

Open and Save PDF file in C# and VB.NET

Opening an existing PDF document

You can open an existing PDF document by using the [PdfLoadedDocument](#) class. The following example shows how to load an existing document from physical path.

C#

```

//Open an existing document from file system
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");

```

VB.NET

```

'Open an existing document from file system
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")

```

UWP

```

//PDF supports opening an existing PDF document from physical path only in
Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.

```

ASP.NET CORE

```

//PDF supports opening an existing PDF document from physical path only in
Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.

```

XAMARIN

```
//PDF supports opening an existing PDF document from physical path only in  
Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

Opening an existing PDF document from Stream

You can open an existing document from stream by using [PdfLoadedDocument](#) class as shown below.

C#

```
//Opens an existing document from stream  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(stream);
```

VB.NET

```
'Opens an existing document from stream  
Dim loadedDocument As New PdfLoadedDocument(stream)
```

UWP

```
//Opens an existing document from stream  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(stream);
```

ASP.NET CORE

```
//Opens an existing document from stream  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(stream);
```

XAMARIN

```
//Opens an existing document from stream  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(stream);
```

Opening an existing PDF document from byte array

You can open an existing document from byte array by using [PdfLoadedDocument](#) class as shown in the below code snippet.

C#

```
//Open an existing document from byte array  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(byteArray);
```

VB.NET

```
'Opens an existing document from byte array  
Dim loadedDocument As New PdfLoadedDocument(byteArray)
```

UWP

```
//Open an existing document from byte array  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(byteArray);
```

ASP.NET CORE

```
//Open an existing document from byte array  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(byteArray);
```

XAMARIN

```
//Open an existing document from byte array  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(byteArray);
```

Opening an Encrypted PDF document

You can open an existing encrypted PDF document from either the file system or the stream or the byte array using the following overloads as shown below

C#

```
//Open an existing encrypted document from disk.  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf",  
"password");
```

VB.NET

```
'Open an existing encrypted document from disk.  
Dim loadedDocument As New PdfLoadedDocument("Input.pdf", "password")
```

UWP

```
//PDF supports opening an Encrypted PDF document from physical path only in  
Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//PDF supports opening an Encrypted PDF document from physical path only in  
Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

XAMARIN

```
//PDF supports opening an Encrypted PDF document from physical path only in  
Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

C#

```
//Open an existing encrypted document from stream.  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(stream,  
"password");
```

VB.NET

```
'Open an existing encrypted document from stream.  
Dim loadedDocument As New PdfLoadedDocument(stream, "password")
```

UWP

```
//Open an existing encrypted document from stream.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(stream,
"password");
```

ASP.NET CORE

```
//Open an existing encrypted document from stream.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(stream,
"password");
```

XAMARIN

```
//Open an existing encrypted document from stream.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(stream,
"password");
```

C#

```
//Open an existing encrypted document from byte array.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(byteArray,
"password");
```

VB.NET

```
'Open an existing encrypted document from byte array.
Dim loadedDocument As New PdfLoadedDocument(byteArray, "password")
```

UWP

```
//Open an existing encrypted document from byte array.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(byteArray,
"password");
```

ASP.NET CORE

```
//Open an existing encrypted document from byte array.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(byteArray,
"password");
```

XAMARIN

```
//Open an existing encrypted document from byte array.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(byteArray,
"password");
```

Opening a corrupted PDF document

You can open a corrupted PDF document from either the file system or the stream or the byte array using the following overloads as shown below

C#

```
//Open an existing corrupted document from disk.
```

```
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf", true);
```

VB.NET

```
'Open an existing corrupted document from disk.  
Dim loadedDocument As New PdfLoadedDocument("Input.pdf", True)
```

UWP

```
//PDF supports opening a corrupted PDF document from physical path only in  
Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//PDF supports opening a corrupted PDF document from physical path only in  
Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

XAMARIN

```
//PDF supports opening a corrupted PDF document from physical path only in  
Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

C#

```
//Open an existing corrupted document from stream.  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(stream, true);
```

VB.NET

```
'Open an existing corrupted document from stream.  
Dim loadedDocument As New PdfLoadedDocument(stream, True)
```

UWP

```
//Open an existing corrupted document from stream.  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(stream, true);
```

ASP.NET CORE

```
//Open an existing corrupted document from stream.  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(stream, true);
```

XAMARIN

```
//Open an existing corrupted document from stream.  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(stream, true);
```

C#

```
//Open an existing corrupted document from byte array.  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(byteArray, true);
```

VB.NET

```
'Open an existing corrupted document from byte array.  
Dim loadedDocument As New PdfLoadedDocument(byteArray, True)
```

UWP

```
//Open an existing corrupted document from byte array.  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(byteArray, true);
```

ASP.NET CORE

```
//Open an existing corrupted document from byte array.  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(byteArray, true);
```

XAMARIN

```
//Open an existing corrupted document from byte array.  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(byteArray, true);
```

Note: 1. The OpenAndRepair overload is capable of resolving basic cross reference offset issues and cannot repair complex document corruption.

2. Using this overload may cause performance delay when compared with other overloads, due to the repairing process.

Saving a PDF document to file system

You can save the manipulated PDF document to file system using [Save](#) method of [PdfLoadedDocument](#) class.

C#

```
//Load an existing PDF document  
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");  
//To-Do some manipulation  
//To-Do some manipulation  
//Save the document in file system  
loadedDocument.Save("Output.pdf");
```

VB.NET

```
' Load an existing PDF document  
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")  
'To-Do some manipulation  
'To-Do some manipulation  
'Save the document in file system  
loadedDocument.Save("Output.pdf")
```

UWP

```
//Load the PDF document as stream
```



```

Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(pdfStream);
//To-Do some manipulation
//To-Do some manipulation
//Save the document into stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//To-Do some manipulation
//To-Do some manipulation
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//To-Do some manipulation
//To-Do some manipulation
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);

```

```
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Saving a PDF document to stream

You can also save the manipulated PDF document to stream using overloads of [Save](#) method.

C#

```
//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//To-Do some manipulation
//To-Do some manipulation
//Creates an instance of memory stream
MemoryStream stream = new MemoryStream();
//Save the document stream
loadedDocument.Save(stream);
```

VB.NET

```
' Load an existing PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'To-Do some manipulation
'To-Do some manipulation
'Creates an instance of memory stream
Dim stream As New MemoryStream()
'Save the document to stream
loadedDocument.Save(stream)
```

UWP

```
//Load the PDF document as stream
Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(pdfStream);
//To-Do some manipulation
//To-Do some manipulation
//Save the document into stream.
```

```
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//To-Do some manipulation
//To-Do some manipulation
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//To-Do some manipulation
//To-Do some manipulation
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
```

Saving a PDF document into the same file or stream

You can also resave the manipulated PDF document to the same file using overloads of [Save](#) method.

C#

```
//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//To-Do some manipulation
//To-Do some manipulation
//Resave the document to the same file
loadedDocument.Save();
```

VB.NET

```
' Load an existing PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'To-Do some manipulation
'To-Do some manipulation
'Resave the document to the same file
loadedDocument.Save()
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
```

```
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//To-Do some manipulation
//To-Do some manipulation
//Resave the document to the same file
await loadedDocument.Save();
```

ASP.NET CORE

```
//PDF supports saving a PDF document into the same file only in Windows
Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms.
```

XAMARIN

```
//PDF supports saving a PDF document into the same file only in Windows
Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms.
```

C#

```
//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(stream);
//To-Do some manipulation
//To-Do some manipulation
//Resave the document to the same stream
loadedDocument.Save();
```

VB.NET

```
' Load an existing PDF document
Dim loadedDocument As New PdfLoadedDocument(stream)
'To-Do some manipulation
'To-Do some manipulation
'Resave the document to the same stream
loadedDocument.Save()
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//To-Do some manipulation
//To-Do some manipulation
```

```
//Resave the document to the same file
await loadedDocument.Save();
```

ASP.NET CORE

```
//PDF supports saving a PDF document into the same file only in Windows Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms.
```

XAMARIN

```
//PDF supports saving a PDF document into the same file only in Windows Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms.
```

Closing a document

After the document manipulation and save operation are completed, you should close the instance of [PdfLoadedDocument](#), in order to release all the memory consumed by PDF DOM. The following code snippet illustrates how to close a [PdfLoadedDocument](#) instance.

C#

```
//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//To-Do some manipulation
//To-Do some manipulation
//Save the document in file system
loadedDocument.Save("Output.pdf");
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
' Load an existing PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'To-Do some manipulation
'To-Do some manipulation
'Save the document in file system
loadedDocument.Save("Output.pdf")
'Close the document
loadedDocument.Close(True)
```

UWP

```
//Load the PDF document as stream
Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.Data.Input.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(pdfStream);
//To-Do some manipulation
//To-Do some manipulation
//Save the document into stream.
```

```
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//To-Do some manipulation
//To-Do some manipulation
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//To-Do some manipulation
//To-Do some manipulation
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
```

Note: Close() method will dispose all the memory consumed by PDF DOM.

Close(true) method will dispose all the memory consumed by PDF DOM as well as disposes its document stream

Working with Document

Adding the document settings

Essential PDF supports various page setting options to control the page display, through [PageSettings](#) property of [PdfDocument](#).

You can choose the standard or custom page size when you add a page to the PDF document. This below sample illustrates how to create a PDF document with standard page size.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the page size.
document.PageSettings.Size = PdfPageSize.A4;
//Add a page to the document.
PdfPage page = document.Pages.Add();
```

```

//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
'Set the page size.
document.PageSettings.Size = PdfPageSize.A4
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Set the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
'Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0,
0))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the page size.
document.PageSettings.Size = PdfPageSize.A4;
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the page size.
document.PageSettings.Size = PdfPageSize.A4;
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the page size.
document.PageSettings.Size = PdfPageSize.A4;
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
```



```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

You can create a PDF document with custom page size by using the following code snippet.

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the custom page size.
document.PageSettings.Size = new.SizeF(200,300);
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
'Set the custom page size.
document.PageSettings.Size = New.SizeF(200, 300)
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Set the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
'Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0,
0))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)

```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the custom page size.
document.PageSettings.Size = new SizeF(200, 300);
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the custom page size.
document.PageSettings.Size = new Syncfusion.Drawing.SizeF(200, 300);
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
```

```

PdfDocument document = new PdfDocument();
// Set the custom page size.
document.PageSettings.Size = new Syncfusion.Drawing.SizeF(200, 300);
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

You can change page orientation from portrait to landscape, through [PdfPageOrientation](#) Enum by using the following code snippet.

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the page size.
document.PageSettings.Size = PdfPageSize.A4;
//Change the page orientation to landscape
document.PageSettings.Orientation = PdfPageOrientation.Landscape;
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document.
document.Save("Output.pdf");
//Close the document.

```

```
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
' Set the page size.
document.PageSettings.Size = PdfPageSize.A4
'Change the page orientation to landscape
document.PageSettings.Orientation = PdfPageOrientation.Landscape
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Set the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
'Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0,
0))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the page size.
document.PageSettings.Size = PdfPageSize.A4;
//Change the page orientation to landscape
document.PageSettings.Orientation = PdfPageOrientation.Landscape;
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the page size.
```

```

document.PageSettings.Size = PdfPageSize.A4;
//Change the page orientation to landscape
document.PageSettings.Orientation = PdfPageOrientation.Landscape;
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the page size.
document.PageSettings.Size = PdfPageSize.A4;
//Change the page orientation to landscape
document.PageSettings.Orientation = PdfPageOrientation.Landscape;
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)

```

```
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

You can also change orientation by setting the rotation angle using [PdfPageRotateAngle](#) Enum. The following code snippet illustrates the same.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the page size.
document.PageSettings.Size = PdfPageSize.A4;
//Change the page orientation to 90°
document.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90;
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Set the page size.
document.PageSettings.Size = PdfPageSize.A4
'Change the page orientation to 90°
document.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Set the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
'Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0,
0))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the page size.
document.PageSettings.Size = PdfPageSize.A4;
//Change the page orientation to 90°
document.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90;
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the page size.
document.PageSettings.Size = PdfPageSize.A4;
//Change the page orientation to 90°
document.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90;
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
```

```
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
// Set the page size.
document.PageSettings.Size = PdfPageSize.A4;
//Change the page orientation to 90°
document.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90;
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Creating sections in a PDF

PDF sections are parts of the PDF document, which may contain one or more pages with their unique page settings. The following code snippet illustrates how to create a [PdfSection](#) in a PDF document.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a section to PDF document.
PdfSection section = document.Sections.Add();
//Add pages to the section
PdfPage page = section.Pages.Add();
//Create PDF graphics for the page
```



```

PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
'Add a section to PDF document.
Dim section As PdfSection = document.Sections.Add()
'Add pages to the section
Dim page As PdfPage = section.Pages.Add()
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Set the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
'Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0, 0))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a section to PDF document.
PdfSection section = document.Sections.Add();
//Add pages to the section
PdfPage page = section.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a section to PDF document.
PdfSection section = document.Sections.Add();
//Add pages to the section
PdfPage page = section.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a section to PDF document.
PdfSection section = document.Sections.Add();
//Add pages to the section
PdfPage page = section.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Printing PDF document

To print a PDF document, the following assemblies have to be added as references to the project.

1. Syncfusion.Compression.Base.dll
2. Syncfusion.Pdf.Base.dll
3. Syncfusion.PdfViewer.Windows.dll

The following code snippet illustrates how to print a PDF document.

C#

```

PdfDocumentView viewer = new PdfDocumentView();
//Load the PDF document
viewer.Load("Input.pdf");
//Initialize print dialog.
PrintDialog dialog = new PrintDialog();
dialog.AllowPrintToFile = true;
dialog.AllowSomePages = true;
dialog.AllowCurrentPage = true;
dialog.Document = viewer.PrintDocument;
//Print the PDF document
dialog.Document.Print();
//Dispose the viewer
viewer.Dispose();

```

VB.NET

```

Dim viewer As New PdfDocumentView()
'Load the PDF document
viewer.Load("Input.pdf")
'Initialize print dialog.
Dim dialog As New PrintDialog()
dialog.AllowPrintToFile = True
dialog.AllowSomePages = True
dialog.AllowCurrentPage = True
dialog.Document = viewer.PrintDocument
'Print the PDF document
dialog.Document.Print()
'Dispose the viewer
viewer.Dispose()

```

Working with document properties

Essential PDF allows you to set, read and modify the document information of a PDF like Author, CreationDate, Subject, Title, and Producer etc. The [DocumentInformation](#) property of the [PdfDocument](#) or [PdfLoadedDocument](#) provides access to this information.

The following code snippet illustrates how to set PDF document information.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Set document information.
document.DocumentInformation.Author = "Syncfusion";
document.DocumentInformation.CreationDate = DateTime.Now;
document.DocumentInformation.Creator = "Essential PDF";
document.DocumentInformation.Keywords = "PDF";
document.DocumentInformation.Subject = "Document information DEMO";
document.DocumentInformation.Title = "Essential PDF Sample";
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Set document information.
document.DocumentInformation.Author = "Syncfusion"
document.DocumentInformation.CreationDate = DateTime.Now
document.DocumentInformation.Creator = "Essential PDF"
document.DocumentInformation.Keywords = "PDF"
document.DocumentInformation.Subject = "Document information DEMO"
document.DocumentInformation.Title = "Essential PDF Sample"
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Set the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
'Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0, 0))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Set document information.
document.DocumentInformation.Author = "Syncfusion";
document.DocumentInformation.CreationDate = DateTime.Now;
document.DocumentInformation.Creator = "Essential PDF";
document.DocumentInformation.Keywords = "PDF";
document.DocumentInformation.Subject = "Document information DEMO";
document.DocumentInformation.Title = "Essential PDF Sample";
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Set document information.
document.DocumentInformation.Author = "Syncfusion";
document.DocumentInformation.CreationDate = DateTime.Now;
document.DocumentInformation.Creator = "Essential PDF";
document.DocumentInformation.Keywords = "PDF";
document.DocumentInformation.Subject = "Document information DEMO";
document.DocumentInformation.Title = "Essential PDF Sample";
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
```

```
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Set document information.
document.DocumentInformation.Author = "Syncfusion";
document.DocumentInformation.CreationDate = DateTime.Now;
document.DocumentInformation.Creator = "Essential PDF";
document.DocumentInformation.Keywords = "PDF";
document.DocumentInformation.Subject = "Document information DEMO";
document.DocumentInformation.Title = "Essential PDF Sample";
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

The following code snippet shows how to read and modify the document properties of an existing PDF document.

C#

```
//Create a new PDF document.
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//Modify document information.
document.DocumentInformation.Author = "Syncfusion";
document.DocumentInformation.CreationDate = DateTime.Now;
document.DocumentInformation.Creator = "Essential PDF";
document.DocumentInformation.Keywords = "PDF";
document.DocumentInformation.Subject = "Document information DEMO";
document.DocumentInformation.Title = "Essential PDF Sample";
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfLoadedDocument("Input.pdf")
'Modify document information.
document.DocumentInformation.Author = "Syncfusion"
document.DocumentInformation.CreationDate = DateTime.Now
document.DocumentInformation.Creator = "Essential PDF"
document.DocumentInformation.Keywords = "PDF"
document.DocumentInformation.Subject = "Document information DEMO"
document.DocumentInformation.Title = "Essential PDF Sample"
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await document.OpenAsync(file);
//Modify document information.
document.DocumentInformation.Author = "Syncfusion";
document.DocumentInformation.CreationDate = DateTime.Now;
document.DocumentInformation.Creator = "Essential PDF";
document.DocumentInformation.Keywords = "PDF";
document.DocumentInformation.Subject = "Document information DEMO";
document.DocumentInformation.Title = "Essential PDF Sample";
//Save the document into stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
```

```
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Modify document information.
document.DocumentInformation.Author = "Syncfusion";
document.DocumentInformation.CreationDate = DateTime.Now;
document.DocumentInformation.Creator = "Essential PDF";
document.DocumentInformation.Keywords = "PDF";
document.DocumentInformation.Subject = "Document information DEMO";
document.DocumentInformation.Title = "Essential PDF Sample";
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Modify document information.
document.DocumentInformation.Author = "Syncfusion";
document.DocumentInformation.CreationDate = DateTime.Now;
document.DocumentInformation.Creator = "Essential PDF";
document.DocumentInformation.Keywords = "PDF";
document.DocumentInformation.Subject = "Document information DEMO";
document.DocumentInformation.Title = "Essential PDF Sample";
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
```



```
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Performing incremental update for PDF document

The Essential PDF supports incremental update for PDF document. The content of a PDF file can be updated incrementally without rewriting the entire file. Changes are appended to the end of the file, leaving its original contents intact. The main benefit is small changes to a large PDF document can be saved quickly but the resultant document size gets increased compared with the original PDF document. Disabling the [IncrementalUpdate](#) of [PdfFileStructure](#) will rewrite the entire file, which results in a smaller PDF. This is illustrated in the following code sample.

C#

```
//Load the PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Disable the incremental update
loadedDocument.FileStructure.IncrementalUpdate = false;
//Set the compression level
loadedDocument.Compression = PdfCompressionLevel.Best;
//Save the document
loadedDocument.Save("Output.pdf");
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Disable the incremental update
loadedDocument.FileStructure.IncrementalUpdate = False
'Set the compression level
loadedDocument.Compression = PdfCompressionLevel.Best
'Save the document
loadedDocument.Save("Output.pdf")
'Close the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
```

```

//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Disable the incremental update
loadedDocument.FileStructure.IncrementalUpdate = false;
//Set the compression level
loadedDocument.Compression = PdfCompressionLevel.Best;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open, FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Disable the incremental update
loadedDocument.FileStructure.IncrementalUpdate = false;
//Set the compression level
loadedDocument.Compression = PdfCompressionLevel.Best;
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into stream
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Disable the incremental update
loadedDocument.FileStructure.IncrementalUpdate = false;
//Set the compression level
loadedDocument.Compression = PdfCompressionLevel.Best;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);

```

```
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Choosing the viewer preferences

Essential PDF allows you to set various PDF viewer preferences to be used when the generated PDF document is displayed in a PDF reader application.

You can hide the menu bar and toolbar by enabling [HideMenubar](#) and [HideToolbar](#) properties of [PdfViewerPreferences](#) respectively. This is illustrated in the following code sample.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Hide viewer application's menu bar.
document.ViewerPreferences.HideMenubar = true;
//Hide viewer application's toolbar.
document.ViewerPreferences.HideToolbar = true;
//Shows user interface elements in the document's window (such as scroll
bars and navigation controls).
document.ViewerPreferences.HideWindowUI = false;
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Add a page to the document.
```

```

Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Set the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
'Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0,
0))
'Hides viewer application's menu bar.
document.ViewerPreferences.HideMenubar = True
'Hides viewer application's toolbar.
document.ViewerPreferences.HideToolbar = True
'Shows user interface elements in the document's window (such as scroll bars
and navigation controls).
document.ViewerPreferences.HideWindowUI = False
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Hide viewer application's menu bar.
document.ViewerPreferences.HideMenubar = true;
//Hide viewer application's toolbar.
document.ViewerPreferences.HideToolbar = true;
//Shows user interface elements in the document's window (such as scroll
bars and navigation controls).
document.ViewerPreferences.HideWindowUI = false;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.

```

```

PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Hide viewer application's menu bar.
document.ViewerPreferences.HideMenubar = true;
//Hide viewer application's toolbar.
document.ViewerPreferences.HideToolbar = true;
//Shows user interface elements in the document's window (such as scroll
bars and navigation controls).
document.ViewerPreferences.HideWindowUI = false;
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Hide viewer application's menu bar.
document.ViewerPreferences.HideMenubar = true;
//Hide viewer application's toolbar.
document.ViewerPreferences.HideToolbar = true;
//Shows user interface elements in the document's window (such as scroll
bars and navigation controls).
document.ViewerPreferences.HideWindowUI = false;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file

```

```
//The operation in Save under Xamarin varies between Windows Phone, Android
//and iOS platforms. Please refer PDF/Xamarin section for respective code
//samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

You can also allow the reader application to initially display the bookmarks, thumbnails or attachments using [PdfPageMode](#) Enum.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Show the attachments panel.
document.ViewerPreferences.PageMode = PdfPageMode.UseAttachments;
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Set the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
'Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0,
0))
'Show the attachments panel.
document.ViewerPreferences.PageMode = PdfPageMode.UseAttachments
'Save the document.
document.Save("Output.pdf")
'Close the document.
```

```
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Show the attachments panel.
document.ViewerPreferences.PageMode = PdfPageMode.UseAttachments;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Show the attachments panel.
document.ViewerPreferences.PageMode = PdfPageMode.UseAttachments;
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Show the attachments panel.
document.ViewerPreferences.PageMode = PdfPageMode.UseAttachments;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Adding document action

Please refer to the [actions](#) section for more document level operations using the PdfJavascript and PDF actions.

Working in Multi-Threading Environment

Essential PDF allows you to create or modify PDF documents simultaneously in multi-threading environment using [EnableThreadSafe](#) property of [PdfDocument](#) class.

The following code sample illustrates how to create a PDF document in multi-threading environment.

C#

```

IEnumerable<int> works = Enumerable.Range(0, 100);
Parallel.ForEach(works, index => GeneratePDF(index));
private void GeneratePDF(int index)
{
//Enable the thread safe in PDF document.
PdfDocument.EnableThreadSafe = true;
//Create a new PDF document.

```



```

PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
string name = Guid.NewGuid().ToString();
//Save the document.
document.Save(name+".pdf");
//Close the document.
document.Close(true);
}

```

VB.NET

```

Dim works As IEnumerable(Of Integer) = Enumerable.Range(0, 100)
Parallel.ForEach(works, Sub(index) GeneratePDF(index))
Private Sub GeneratePDF(ByVal index As Integer)
    'Enable the thread safe in PDF document.
    PdfDocument.EnableThreadSafe = True
    'Create a new PDF document.
    Dim document As PdfDocument = New PdfDocument()
    'Add a page to the document.
    Dim page As PdfPage = document.Pages.Add()
    'Create PDF graphics for the page.
    Dim graphics As PdfGraphics = page.Graphics
    'Set the standard font.
    Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
    'Draw the text.
    graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0, 0))
    Dim name As String = Guid.NewGuid().ToString()
    'Save the document.
    document.Save(name + ".pdf")
    'Close the document.
    document.Close(True)
End Sub

```

You can also modify the existing PDF document in multi-threading environment by using the following code snippet.

C#

```

IEnumerable<int> works = Enumerable.Range(0, 100);
Parallel.ForEach(works, index => GeneratePDF(index));
private void GeneratePDF(int index)
{
    //Enable the thread safe in PDF document.
    PdfDocument.EnableThreadSafe = true;
    //Load a PDF document.
    PdfLoadedDocument doc = new PdfLoadedDocument("input.pdf");
    //Get first page from document
}

```

```

PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
string name = Guid.NewGuid().ToString();
//Save the document.
doc.Save(name+".pdf");
//Close the document.
doc.Close(true);
}

```

VB.NET

```

Dim works As IEnumerable(Of Integer) = Enumerable.Range(0, 100)
Parallel.ForEach(works, Sub(index) GeneratePDF(index))
Private Sub GeneratePDF(ByVal index As Integer)
'Enable the thread safe in PDF document.
PdfDocument.EnableThreadSafe = True
'Load a PDF document.
Dim doc As PdfLoadedDocument = New PdfLoadedDocument("input.pdf")
'Get first page from document
Dim page As PdfLoadedPage = doc.Pages(0)
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Set the standard font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
'Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0,
0))
Dim name As String = Guid.NewGuid().ToString()
'Save the document.
doc.Save(name + ".pdf")
'Close the document.
doc.Close(True)
End Sub

```

Uniform Resource Naming in PDF document

The Essential PDF allows you to create a PDF document with proper uniform resource naming by using the [EnableUniqueResourceNaming](#) property available in the [PdfDocument](#) instance. By default, the resource naming is added uniquely. Disabling the `EnableUniqueResourceNaming` property will create a PDF document with uniform resource names.

The following code snippet explains how to have uniform resource naming in a PDF document.

C#

```

//Disable unique resource naming
PdfDocument.EnableUniqueResourceNaming = false;
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document

```

```

PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new instance for PDF font
PdfFont font1 = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text
graphics.DrawString("Hello World!!!", font1, PdfBrushes.Blue, new PointF(50,
50));
//Create new instance for PDF font
PdfFont font2 = new PdfTrueTypeFont(new Font("Arial", 20), true);
//Draw the text
graphics.DrawString("Hello World!!!", font2, PdfBrushes.Blue, new PointF(50,
100));
//Create new instance for PDF font
PdfFont font3 = new PdfCjkStandardFont(PdfCjkFontFamily.HeiseiMinchoW3, 20);
//Draw the text
graphics.DrawString("こんにちは世界", font3, PdfBrushes.Blue, new PointF(50,
150));
//Save and close the document
doc.Save("Output.pdf");
doc.Close(true);

```

VB.NET

```

'Disable unique resource naming
PdfDocument.EnableUniqueResourceNaming = False
'Create a new PDF document
Dim doc As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = doc.Pages.Add
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create new instance for PDF font
Dim font1 As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
'Draw the text
graphics.DrawString("Hello World!!!", font1, PdfBrushes.Blue, New PointF(50,
50))
'Create new instance for PDF font
Dim font2 As PdfFont = New PdfTrueTypeFont(New Font("Arial", 20), True)
'Draw the text
graphics.DrawString("Hello World!!!", font2, PdfBrushes.Blue, New PointF(50,
100))
'Create new instance for PDF font
Dim font3 As PdfFont = New
PdfCjkStandardFont(PdfCjkFontFamily.HeiseiMinchoW3, 20)
'Draw the text
graphics.DrawString("こんにちは世界", font3, PdfBrushes.Blue, New PointF(50,
150))
'Save and close the document
doc.Save("Output.pdf")
doc.Close(True)

```

UWP

```
//Disable unique resource naming
```

```

PdfDocument.EnableUniqueResourceNaming = false;
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new instance for PDF font
PdfFont font1 = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text
graphics.DrawString("Hello World!!!", font1, PdfBrushes.Blue, new PointF(50,
50));
//Create new instance for PDF font
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Arial.ttf");
PdfFont font2 = new PdfTrueTypeFont(fontStream, 20);
//Draw the text
graphics.DrawString("Hello World!!!", font2, PdfBrushes.Blue, new PointF(50,
100));
//Create new instance for PDF font
PdfFont font3 = new PdfCjkStandardFont(PdfCjkFontFamily.HeiseiMinchoW3, 20);
//Draw the text
graphics.DrawString("こんにちは世界", font3, PdfBrushes.Blue, new PointF(50,
150));
//Create memory stream
MemoryStream ms = new MemoryStream();
//Open the document in browser after saving it
doc.Save(ms);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respective code samples
Save(ms, "Output.pdf");

```

ASP.NET CORE

```

//Disable unique resource naming
PdfDocument.EnableUniqueResourceNaming = false;
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new instance for PDF font
PdfFont font1 = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text
graphics.DrawString("Hello World!!!", font1, PdfBrushes.Blue, new PointF(50,
50));
//Create new instance for PDF font
FileStream fontStream = new FileStream("Arial.ttf", FileMode.Open,
FileAccess.Read);
PdfFont font2 = new PdfTrueTypeFont(fontStream, 20);
//Draw the text

```

```

graphics.DrawString("Hello World!!!", font2, PdfBrushes.Blue, new PointF(50,
100));
//Create new instance for PDF font
PdfFont font3 = new PdfCjkStandardFont(PdfCjkFontFamily.HeiseiMinchoW3, 20);
//Draw the text
graphics.DrawString("こんにちは世界", font3, PdfBrushes.Blue, new PointF(50,
150));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileName = "Output.pdf";
return fileStreamResult;

```

XAMARIN

```

//Disable unique resource naming
PdfDocument.EnableUniqueResourceNaming = false;
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new instance for PDF font
PdfFont font1 = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text
graphics.DrawString("Hello World!!!", font1, PdfBrushes.Blue, new PointF(50,
50));
//Create new instance for PDF font
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Arial.ttf");
PdfFont font2 = new PdfTrueTypeFont(fontStream, 20);
//Draw the text
graphics.DrawString("Hello World!!!", font2, PdfBrushes.Blue, new PointF(50,
100));
//Create new instance for PDF font
PdfFont font3 = new PdfCjkStandardFont(PdfCjkFontFamily.HeiseiMinchoW3, 20);
//Draw the text
graphics.DrawString("こんにちは世界", font3, PdfBrushes.Blue, new PointF(50,
150));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Memory Optimization

Essential PDF provides support for optimization of memory using [EnableMemoryOptimization](#) property in [PdfDocument](#) instance. Optimization will be effective only with merge, append and import functions.

Enabling this property will optimize the memory but difference in time occurs based on the document size. This is illustrated in the following code sample.

C#

```

//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("file1.pdf");
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Enable memory optimization
document.EnableMemoryOptimization = true;
//Append the document with source document
document.Append(loadedDocument);
//Save the PDF document
document.Save("Output.pdf");
//Close the documents
document.Close(true);
loadedDocument.Close(true);

```

VB.NET

```

'Load an existing PDF document
Dim loadedDocument As New PdfLoadedDocument("file1.pdf")
'Create a new PDF document
Dim document As New PdfDocument()
'Enable memory optimization
document.EnableMemoryOptimization = True
'Append the document with source document
document.Append(loadedDocument)
'Save the PDF document
document.Save("Output.pdf")
'Close the documents
document.Close(True)
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();

```

```

picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Create an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Load an existing PDF document through Open method of PdfLoadedDocument
class
await loadedDocument.OpenAsync(file);
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Enable memory optimization
document.EnableMemoryOptimization = true;
//Append the document with source document
document.Append(loadedDocument);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
Await document.SaveAsync(stream);
//Close the documents
document.Close(true);
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load an existing PDF document
FileStream docStream = new FileStream("file1.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Enable memory optimization
document.EnableMemoryOptimization = true;
//Append the document with source document
document.Append(loadedDocument);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents
document.Close(true);
loadedDocument.Close(true);
//Defining the content type for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream

```

```

Stream docStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
    file1.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Enable memory optimization
document.EnableMemoryOptimization = true;
//Append the document with source document
document.Append(loadedDocument);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the documents
document.Close(true);
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
    "application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
    "application/pdf", stream);
}

```

Find corrupted PDF document

Syncfusion PDF Library provides support to check whether the existing PDF document is corrupted or not with corruption details. The following code snippet explains how to find the corrupted PDF document.

C#

```

//Create a new instance for the PDF analyzer
PdfDocumentAnalyzer analyzer = new PdfDocumentAnalyzer("Input.pdf");
//Get the syntax errors
SyntaxAnalyzerResult result = analyzer.AnalyzeSyntax();
//Check whether the document is corrupted or not
if (result.IsCorrupted)
{
    //Get syntax error details from results.error
    StringBuilder builder = new StringBuilder();
    builder.AppendLine("The PDF document is corrupted.");
    int count = 1;
    foreach (PdfException exception in result.Errors)
    {
        builder.AppendLine(count++.ToString() + ": " + exception.Message);
    }
}
else
{

```



```
//No syntax error found in the provided PDF document
}
analyzer.Close();
```

VB.NET

```
'Create a new instance for the PDF analyzer
Dim analyzer As PdfDocumentAnalyzer = New PdfDocumentAnalyzer("Input.pdf")
'Get the syntax errors
Dim result As SyntaxAnalyzerResult = analyzer.AnalyzeSyntax
'Check whether the document is corrupted or not
If result.IsCorrupted Then
'Get syntax error details from results.error
builder = New StringBuilder
builder.AppendLine("The PDF document is corrupted.")
Dim count = 1
For Each exception As PdfException In result.Errors
builder.AppendLine(Math.Min(Threading.Interlocked.Increment(count), count -
1).ToString & ": " & exception.Message)
Next
Else
'No syntax error found in the provided PDF document.
End If
analyzer.Close()
```

UWP

```
//Load the PDF document as stream
Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.pdf");
//Create a new instance for the PDF analyzer
PdfDocumentAnalyzer analyzer = new PdfDocumentAnalyzer(pdfStream);
//Get the syntax errors.
SyntaxAnalyzerResult result = analyzer.AnalyzeSyntax();
//Check whether the document is corrupted or not
if (result.IsCorrupted)
{
//Get syntax error details from results.error
StringBuilder builder = new StringBuilder();
builder.AppendLine("The PDF document is corrupted.");
int count = 1;
foreach (PdfException exception in result.Errors)
{
builder.AppendLine(count++.ToString() + ": " + exception.Message);
}
}
else
{
//No syntax error found in the provided PDF document
}
analyzer.Close();
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
//Create a new instance for the PDF analyzer
PdfDocumentAnalyzer analyzer = new PdfDocumentAnalyzer(docStream);
//Get the syntax errors
SyntaxAnalyzerResult result = analyzer.AnalyzeSyntax();
//Check whether the document is corrupted or not
if (result.IsCorrupted)
{
//Get syntax error details from results.error
StringBuilder builder = new StringBuilder();
builder.AppendLine("The PDF document is corrupted.");
int count = 1;
foreach (PdfException exception in result.Errors)
{
builder.AppendLine(count++.ToString() + ": " + exception.Message);
}
}
else
{
//No syntax error found in the provided PDF document
}
analyzer.Close();
```

XAMARIN

```
//Load the PDF document as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
//Create a new instance for the PDF analyzer
PdfDocumentAnalyzer analyzer = new PdfDocumentAnalyzer(docStream);
//Get the syntax errors
SyntaxAnalyzerResult result = analyzer.AnalyzeSyntax();
//Check whether the document is corrupted or not
if (result.IsCorrupted)
{
//Get syntax error details from results.error
StringBuilder builder = new StringBuilder();
builder.AppendLine("The PDF document is corrupted.");
int count = 1;
foreach (PdfException exception in result.Errors)
{
builder.AppendLine(count++.ToString() + ": " + exception.Message);
}
}
else
{
//No syntax error found in the provided PDF document
}
analyzer.Close();
```

Working with Pages

Adding a new page to the document

The following code sample explains you on how to add a [PdfPage](#) in a PDF document. When multiple pages are added, the new page is always added to the end of the document.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Create a solid brush.
PdfBrush brush = new PdfSolidBrush(Color.Black);
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Draw the text.
graphics.DrawString("Hello world!", font, brush, new PointF(20, 20));
//Save the document.
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Add a page.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Create a solid brush.
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
'Set the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 14)
'Draw the text.
graphics.DrawString("Hello world!", font, brush, New PointF(20, 20))
'Save the document.
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Create a solid brush.
PdfBrush brush = new PdfSolidBrush(new PdfColor(0, 0, 0));
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Draw the text.
graphics.DrawString("Hello world!", font, brush, new PointF(20, 20));
```

```
//Save the document as stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Create a solid brush.
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Black);
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Draw the text.
graphics.DrawString("Hello world!", font, brush, new
Syncfusion.Drawing.PointF(20, 20));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Create a solid brush.
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Black);
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Draw the text.
graphics.DrawString("Hello world!", font, brush, new
Syncfusion.Drawing.PointF(20, 20));
//Save the document as stream.
```

```

MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
        "application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
        "application/pdf", stream);
}

```

Inserting pages in a document

You can insert an empty page at any location in the existing PDF document using [Insert](#) method. The below code snippet explains the same.

C#

```

//Load the PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Insert a new page in the beginning of the document
loadedDocument.Pages.Insert(0);
//Save and close the document
loadedDocument.Save("Output.pdf");
loadedDocument.Close(true);

```

VB.NET

```

'Load the PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Insert a new page in the beginning of the document
loadedDocument.Pages.Insert(0)
'Save and close the document
loadedDocument.Save("Output.pdf")
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class

```

```

await loadedDocument.OpenAsync(file);
//Insert a new page in the beginning of the document
loadedDocument.Pages.Insert(0);
//Save the document as stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Insert a new page in the beginning of the document
loadedDocument.Pages.Insert(0);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Insert a new page in the beginning of the document
loadedDocument.Pages.Insert(0);
//Save the document as stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)

```

```
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Adding margin to the PDF pages

You can add margin to all the PDF pages of the PDF document using the [PageSettings](#) property. The following code snippet illustrates the same.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Set margin for all the pages
document.PageSettings.Margins.All = 10;
//Add a page.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Creates a solid brush.
PdfBrush brush = new PdfSolidBrush(Color.Black);
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Draw the text.
graphics.DrawString("Hello world!", font, brush, new PointF(20, 20));
//Save and close the document.
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Set margin for all the pages
document.PageSettings.Margins.All = 10
'Add a page.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Create a solid brush.
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
'Set the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 14)
'Draw the text.
graphics.DrawString("Hello world!", font, brush, New PointF(20, 20))
'Save the document.
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Set margin for all the pages
document.PageSettings.Margins.All = 10;
//Add a page.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Creates a solid brush.
PdfBrush brush = new PdfSolidBrush(new PdfColor(0, 0, 0));
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Draw the text.
graphics.DrawString("Hello world!", font, brush, new PointF(20, 20));
//Save the document as stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Set margin for all the pages
document.PageSettings.Margins.All = 10;
//Add a page.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Creates a solid brush.
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Black);
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Draw the text.
graphics.DrawString("Hello world!", font, brush, new
Syncfusion.Drawing.PointF(20, 20));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
```



```
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Set margin for all the pages
document.PageSettings.Margins.All = 10;
//Add a page.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Creates a solid brush.
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Black);
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Draw the text.
graphics.DrawString("Hello world!", font, brush, new
Syncfusion.Drawing.PointF(20, 20));
//Save the document as stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Adding sections with different page settings

Essential PDF supports adding sections with different page settings like [Height](#), [Margins](#), [Orientation](#), [Rotate](#), [Size](#), [Transition](#) and [Width](#). You can add sections to a PDF document by using the [PdfSection](#) available in [PdfDocument](#) instance and create page settings to the PdfSection using the [PageSettings](#) property.

The following code snippet explains how to add more sections to a PDF document with different page settings.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create a solid brush and standard font
PdfBrush brush = new PdfSolidBrush(Color.Black);
```

```
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);  
//Section - 1  
//Add new section to the document  
PdfSection section = document.Sections.Add();  
//Create page settings to the section  
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle0;  
section.PageSettings.Size = PdfPageSize.A5;  
section.PageSettings.Width = 300;  
section.PageSettings.Height = 400;  
//Add page to the section and initialize graphics for the page  
PdfPage page = section.Pages.Add();  
PdfGraphics graphics = page.Graphics;  
//Draw simple text on the page  
graphics.DrawString("Rotated by 0 degrees", font, brush, new PointF(20,  
20));  
//Section - 2  
//Add new section to the document  
section = document.Sections.Add();  
//Create page settings to the section  
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90;  
section.PageSettings.Width = 300;  
section.PageSettings.Height = 400;  
//Add page to the section and initialize graphics for the page  
page = section.Pages.Add();  
graphics = page.Graphics;  
//Draw simple text on the page  
graphics.DrawString("Rotated by 90 degrees", font, brush, new PointF(20,  
20));  
//Section - 3  
//Add new section to the document  
section = document.Sections.Add();  
//Create page settings to the section  
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle180;  
section.PageSettings.Width = 500;  
section.PageSettings.Height = 200;  
//Add page to the section and initialize graphics for the page  
page = section.Pages.Add();  
graphics = page.Graphics;  
//Draw simple text on the page  
graphics.DrawString("Rotated by 180 degrees", font, brush, new PointF(20,  
20));  
//Section - 4  
//Add new section to the document  
section = document.Sections.Add();  
//Create page settings to the section  
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle270;  
section.PageSettings.Width = 300;  
section.PageSettings.Height = 200;  
//Add page to the section and initialize graphics for the page  
page = section.Pages.Add();  
graphics = page.Graphics;  
//Draw simple text on the page  
graphics.DrawString("Rotated by 270 degrees", font, brush, new PointF(20,  
20));  
//Save the document  
document.Save("Output.pdf");  
//Close the instance of PdfDocument
```

```
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Create a solid brush and standard font
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 14)
'Section - 1
'Add new section to the document
Dim section As PdfSection = document.Sections.Add
'Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle0
section.PageSettings.Size = PdfPageSize.A5
section.PageSettings.Width = 300
section.PageSettings.Height = 400
'Add page to the section and initialize graphics for the page
Dim page As PdfPage = section.Pages.Add
Dim graphics As PdfGraphics = page.Graphics
'Draw simple text on the page
graphics.DrawString("Rotated by 0 degrees", font, brush, New PointF(20, 20))
'Section - 2
'Add new section to the document
section = document.Sections.Add
'Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90
section.PageSettings.Width = 300
section.PageSettings.Height = 400
'Add page to the section and initialize graphics for the page
page = section.Pages.Add
graphics = page.Graphics
'Draw simple text on the page
graphics.DrawString("Rotated by 90 degrees", font, brush, New PointF(20, 20))
'Section - 3
'Add new section to the document
section = document.Sections.Add
'Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle180
section.PageSettings.Width = 500
section.PageSettings.Height = 200
'Add page to the section and initialize graphics for the page
page = section.Pages.Add
graphics = page.Graphics
'Draw simple text on the page
graphics.DrawString("Rotated by 180 degrees", font, brush, New PointF(20, 20))
'Section - 4
'Add new section to the document
section = document.Sections.Add
'Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle270
section.PageSettings.Width = 300
section.PageSettings.Height = 200
'Add page to the section and initialize graphics for the page
```

```

page = section.Pages.Add
graphics = page.Graphics
'Draw simple text on the page
graphics.DrawString("Rotated by 270 degrees", font, brush, New PointF(20,
20))
'Save the document
document.Save("Output.pdf")
'Close the instance of PdfDocument
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create a solid brush and standard font
PdfBrush brush = new PdfSolidBrush(Color.FromArgb(255, 0, 0, 0));
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Section - 1
//Add new section to the document
PdfSection section = document.Sections.Add();
//Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle0;
section.PageSettings.Size = PdfPageSize.A5;
section.PageSettings.Width = 300;
section.PageSettings.Height = 400;
//Add page to the section and initialize graphics for the page
PdfPage page = section.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Draw simple text on the page
graphics.DrawString("Rotated by 0 degrees", font, brush, new PointF(20,
20));
//Section - 2
//Add new section to the document
section = document.Sections.Add();
//Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90;
section.PageSettings.Width = 300;
section.PageSettings.Height = 400;
//Add page to the section and initialize graphics for the page
page = section.Pages.Add();
graphics = page.Graphics;
//Draw simple text on the page
graphics.DrawString("Rotated by 90 degrees", font, brush, new PointF(20,
20));
//Section - 3
//Add new section to the document
section = document.Sections.Add();
//Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle180;
section.PageSettings.Width = 500;
section.PageSettings.Height = 200;
//Add page to the section and initialize graphics for the page
page = section.Pages.Add();
graphics = page.Graphics;
//Draw simple text on the page

```

```

graphics.DrawString("Rotated by 180 degrees", font, brush, new PointF(20,
20));
//Section - 4
//Add new section to the document
section = document.Sections.Add();
//Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle270;
section.PageSettings.Width = 300;
section.PageSettings.Height = 200;
//Add page to the section and initialize graphics for the page
page = section.Pages.Add();
graphics = page.Graphics;
//Draw simple text on the page
graphics.DrawString("Rotated by 270 degrees", font, brush, new PointF(20,
20));
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
document.Save(stream);
//Close the document
document.Close(true);

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create a solid brush and standard font
PdfBrush brush = new PdfSolidBrush(Color.Black);
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Section - 1
//Add new section to the document
PdfSection section = document.Sections.Add();
//Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle0;
section.PageSettings.Size = PdfPageSize.A5;
section.PageSettings.Width = 300;
section.PageSettings.Height = 400;
//Add page to the section and initialize graphics for the page
PdfPage page = section.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Draw simple text on the page
graphics.DrawString("Rotated by 0 degrees", font, brush, new PointF(20,
20));
//Section - 2
//Add new section to the document
section = document.Sections.Add();
//Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90;
section.PageSettings.Width = 300;
section.PageSettings.Height = 400;
//Add page to the section and initialize graphics for the page
page = section.Pages.Add();
graphics = page.Graphics;
//Draw simple text on the page
graphics.DrawString("Rotated by 90 degrees", font, brush, new PointF(20,
20));

```

```
//Section - 3
//Add new section to the document
section = document.Sections.Add();
//Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle180;
section.PageSettings.Width = 500;
section.PageSettings.Height = 200;
//Add page to the section and initialize graphics for the page
page = section.Pages.Add();
graphics = page.Graphics;
//Draw simple text on the page
graphics.DrawString("Rotated by 180 degrees", font, brush, new PointF(20,
20));
//Section - 4
//Add new section to the document
section = document.Sections.Add();
//Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle270;
section.PageSettings.Width = 300;
section.PageSettings.Height = 200;
//Add page to the section and initialize graphics for the page
page = section.Pages.Add();
graphics = page.Graphics;
//Draw simple text on the page
graphics.DrawString("Rotated by 270 degrees", font, brush, new PointF(20,
20));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create a solid brush and standard font
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.FromArgb(255, 0,
0, 0));
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Section - 1
//Add new section to the document
PdfSection section = document.Sections.Add();
//Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle0;
section.PageSettings.Size = PdfPageSize.A5;
section.PageSettings.Width = 300;
section.PageSettings.Height = 400;
//Add page to the section and initialize graphics for the page
PdfPage page = section.Pages.Add();
PdfGraphics graphics = page.Graphics;
```

```

//Draw simple text on the page
graphics.DrawString("Rotated by 0 degrees", font, brush, new PointF(20,
20));
//Section - 2
//Add new section to the document
section = document.Sections.Add();
//Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90;
section.PageSettings.Width = 300;
section.PageSettings.Height = 400;
//Add page to the section and initialize graphics for the page
page = section.Pages.Add();
graphics = page.Graphics;
//Draw simple text on the page
graphics.DrawString("Rotated by 90 degrees", font, brush, new PointF(20,
20));
//Section - 3
//Add new section to the document
section = document.Sections.Add();
//Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle180;
section.PageSettings.Width = 500;
section.PageSettings.Height = 200;
//Add page to the section and initialize graphics for the page
page = section.Pages.Add();
graphics = page.Graphics;
//Draw simple text on the page
graphics.DrawString("Rotated by 180 degrees", font, brush, new PointF(20,
20));
//Section - 4
//Add new section to the document
section = document.Sections.Add();
//Create page settings to the section
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle270;
section.PageSettings.Width = 300;
section.PageSettings.Height = 200;
//Add page to the section and initialize graphics for the page
page = section.Pages.Add();
graphics = page.Graphics;
//Draw simple text on the page
graphics.DrawString("Rotated by 270 degrees", font, brush, new PointF(20,
20));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}

```

```
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
    "application/pdf", stream);
}
```

Get number of pages from a PDF document

You can get page count from the existing PDF document as shown in the following code snippet.

C#

```
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get the page count.
int pageCount = loadedDocument.Pages.Count;
//Close the document.
loadedDocument.Close(true);
```

VB.NET

```
'Load the PDF document.
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the page count.
Dim pageCount As Integer = loadedDocument.Pages.Count
'Close the document.
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the page count.
int pageCount = loadedDocument.Pages.Count;
//Close the document.
loadedDocument.Close(true);
```

ASP.NET CORE

```
//Load the PDF document.
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the page count.
int pageCount = loadedDocument.Pages.Count;
//Close the document.
loadedDocument.Close(true);
```


XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf ");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the page count.
int pageCount = loadedDocument.Pages.Count;
//Close the document.
loadedDocument.Close(true);
```

Importing pages from an existing document.

Essential PDF allows you to import a page or import a range of pages from one document to the other. The following code sample illustrates how to import a range of pages from an existing document using [ImportPageRange](#) method.

C#

```
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Create a new PDF document.
PdfDocument document = new PdfDocument();
int startIndex = 0;
int endIndex = loadedDocument.Pages.Count - 1;
//Import all the pages to the new PDF document.
document.ImportPageRange(loadedDocument, startIndex, endIndex);
//Save the document.
document.Save("Output.pdf");
//Close both document instances.
loadedDocument.Close(true);
document.Close(true);
```

VB.NET

```
'Load the PDF document.
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Create a new PDF document.
Dim document As New PdfDocument()
Dim startIndex As Integer = 0
Dim endIndex As Integer = loadedDocument.Pages.Count - 1
'Import all the pages to the new PDF document.
document.ImportPageRange(loadedDocument, startIndex, endIndex)
'Save the document.
document.Save("Output.pdf")
'Close both document instances.
loadedDocument.Close(True)
document.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
```

```

picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Create a new PDF document.
PdfDocument document = new PdfDocument();
int startIndex = 0;
int endIndex = loadedDocument.Pages.Count - 1;
//Import all the pages to the new PDF document.
document.ImportPageRange(loadedDocument, startIndex, endIndex);
//Save the document as stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document instances.
loadedDocument.Close(true);
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document.
FileStream docStream = new FileStream("Input.pdf", FileMode.Open, FileAccess.Read);
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create a new PDF document.
PdfDocument document = new PdfDocument();
int startIndex = 0;
int endIndex = loadedDocument.Pages.Count - 1;
//Import all the pages to the new PDF document.
document.ImportPageRange(loadedDocument, startIndex, endIndex);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document instances.
document.Close(true);
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create a new PDF document.
PdfDocument document = new PdfDocument();
int startIndex = 0;
int endIndex = loadedDocument.Pages.Count - 1;
//Import all the pages to the new PDF document.
document.ImportPageRange(loadedDocument, startIndex, endIndex);
//Save the document as stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document instances.
document.Close(true);
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Importing pages from an existing document without bookmarks.

You can import a page or range of pages from one document to other without bookmarks. Refer to the following code sample.

Note: Performance will be effective only in the large PDF document.

C#

```

//Load the PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Create the new PDF document
PdfDocument document = new PdfDocument();
int startIndex = 0;
int endIndex = loadedDocument.Pages.Count - 1;
//Import all the pages to the new PDF document without bookmarks
document.ImportPageRange(loadedDocument, startIndex, endIndex, false);
//Save the document
document.Save("Output.pdf");
//Close both document instances
loadedDocument.Close(true);
document.Close(true);
System.Diagnostics.Process.Start("Output.pdf");

```

VB.NET

```

'Load the PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Create the new PDF document
Dim document As PdfDocument = New PdfDocument()
Dim startIndex As Integer = 0
Dim endIndex As Integer = loadedDocument.Pages.Count - 1
'Import all the pages to the new PDF document without bookmarks
document.ImportPageRange(loadedDocument, startIndex, endIndex, False)
'Save the document
document.Save("Output.pdf")
'Close both the document instances
loadedDocument.Close(True)
document.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Create the new PDF document
PdfDocument document = new PdfDocument();
int startIndex = 0;
int endIndex = loadedDocument.Pages.Count - 1;
//Import all the pages to the new PDF document
document.ImportPageRange(loadedDocument, startIndex, endIndex, false);
//Save the document as stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document instances
loadedDocument.Close(true);
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
//Load the PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the new PDF document
PdfDocument document = new PdfDocument();
int startIndex = 0;
int endIndex = loadedDocument.Pages.Count - 1;
//Import all the pages to the new PDF document

```

```

document.ImportPageRange(loaderDocument, startIndex, endIndex, false);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document instances
document.Close(true);
loaderDocument.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoaderDocument loaderDocument = new PdfLoaderDocument(docStream);
//Create the new PDF document
PdfDocument document = new PdfDocument();
int startIndex = 0;
int endIndex = loaderDocument.Pages.Count - 1;
//Import all the pages to the new PDF document
document.ImportPageRange(loaderDocument, startIndex, endIndex, false);
//Save the document as stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document instances
document.Close(true);
loaderDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Rearranging pages in an existing document

You can rearrange the pages in an existing PDF document using [ReArrange](#) method. This method uses zero based start index. The following code snippet illustrates the same.

C#

```
//Load the PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Rearrange the page by index
loadedDocument.Pages.ReArrange(new int[] {1, 0});
//Save and close the document
loadedDocument.Save("Output.pdf");
loadedDocument.Close(true);
```

VB.NET

```
'Load the PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Rearrange the page by index
loadedDocument.Pages.ReArrange(New Integer() {1, 0})
'Save and close the document
loadedDocument.Save("Output.pdf")
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Rearrange the page by index
loadedDocument.Pages.ReArrange(new int[] { 1, 0 });
//Save the document as stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document.
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Rearrange the page by index
loadedDocument.Pages.ReArrange(new int[] { 1, 0 });
```

```
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Rearrange the page by index
loadedDocument.Pages.ReArrange(new int[] { 1, 0 });
//Save the document as stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Changing the page numbers in a PDF document

You can alter the page label for the existing PDF document using [PdfPageLabel](#) class. Refer to the following code snippet.

C#

```
//Load the PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Create a page label
```

```

PdfPageLabel pageLabel = new PdfPageLabel();
//Set the number style with upper case roman letters
pageLabel.NumberStyle = PdfNumberStyle.UpperRoman;
//Set the starting number as 1
pageLabel.StartNumber = 1;
loadedDocument.LoadedPageLabel = pageLabel;
//Save the document
loadedDocument.Save("Output.pdf");
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Create a page label
Dim pageLabel As New PdfPageLabel()
'Set the number style with upper case roman letters
pageLabel.NumberStyle = PdfNumberStyle.UpperRoman
'Set the starting number as 1
pageLabel.StartNumber = 1
loadedDocument.LoadedPageLabel = pageLabel
'Save the document
loadedDocument.Save("Output.pdf")
'Close the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose a file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of the
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
// Create a page label
PdfPageLabel pageLabel = new PdfPageLabel();
//Set the number style with upper case roman letters
pageLabel.NumberStyle = PdfNumberStyle.UpperRoman;
//Set the starting number as 1
pageLabel.StartNumber = 1;
loadedDocument.LoadedPageLabel = pageLabel;
//Save the document as stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document instances
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```


ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
//Load the PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
// Create a page label
PdfPageLabel pageLabel = new PdfPageLabel();
//Set the number style with upper case roman letters
pageLabel.NumberStyle = PdfNumberStyle.UpperRoman;
//Set the starting number as 1
pageLabel.StartNumber = 1;
loadedDocument.LoadedPageLabel = pageLabel;
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
loadedDocument.Save(stream);
//If the position is not set to '0', then the PDF will be empty.
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
// Create a page label
PdfPageLabel pageLabel = new PdfPageLabel();
//Set the number style with upper case roman letters
pageLabel.NumberStyle = PdfNumberStyle.UpperRoman;
//Set the starting number as 1
pageLabel.StartNumber = 1;
loadedDocument.LoadedPageLabel = pageLabel;
//Save the document as stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document instances
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Removing pages from a document

You can remove the pages from the existing PDF document using [RemoveAt](#) method as shown in the below code snippet.

C#

```
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Remove the first page in the PDF document
loadedDocument.Pages.RemoveAt(0);
//Save the document.
loadedDocument.Save("Output.pdf");
//Close the document.
loadedDocument.Close(true);
```

VB.NET

```
'Load the PDF document.
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Remove the first page in the PDF document
loadedDocument.Pages.RemoveAt(0)
'Save the document.
loadedDocument.Save("Output.pdf")
'Close the document.
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Remove the first page in the PDF document
loadedDocument.Pages.RemoveAt(0);
//Save the document as stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
```

```
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document.
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Remove the first page in the PDF document
loadedDocument.Pages.RemoveAt(0);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Remove the first page in the PDF document
loadedDocument.Pages.RemoveAt(0);
//Save the document as stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Rotating a PDF page

You can rotate a particular PDF page in the PDF document using [PdfPageRotateAngle](#) Enum as shown the following code snippet.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a section.
PdfSection section = document.Sections.Add();
//Rotate a section/page
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90;
PdfPage page = section.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Create a solid brush.
PdfBrush brush = new PdfSolidBrush(Color.Black);
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Draws the text.
graphics.DrawString("Rotated by 90 degree", font, brush, new PointF(20,
20));
//Save the document.
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Add a section.
Dim section As PdfSection = document.Sections.Add()
'Rotate a section/page
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90
Dim page As PdfPage = section.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Create a solid brush.
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
'Set the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 14)
'Draw the text.
graphics.DrawString("Rotated by 90 degree", font, brush, New PointF(20, 20))
'Save the document.
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
```

```

//Add a section.
PdfSection section = document.Sections.Add();
//Rotate a section/page
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90;
PdfPage page = section.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Create a solid brush.
PdfBrush brush = new PdfSolidBrush(new PdfColor(0, 0, 0));
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Draws the text.
graphics.DrawString("Rotated by 90 degree", font, brush, new PointF(20,
20));
//Save the document as stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a section.
PdfSection section = document.Sections.Add();
//Rotate a section/page
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90;
PdfPage page = section.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Create a solid brush.
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Black);
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Draws the text.
graphics.DrawString("Rotated by 90 degree", font, brush, new
Syncfusion.Drawing.PointF(20, 20));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a section.
PdfSection section = document.Sections.Add();
//Rotate a section/page
section.PageSettings.Rotate = PdfPageRotateAngle.RotateAngle90;
PdfPage page = section.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Create a solid brush.
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Black);
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Draws the text.
graphics.DrawString("Rotated by 90 degree", font, brush, new
Syncfusion.Drawing.PointF(20, 20));
//Save the document as stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Rotating an existing PDF page

You can also rotate a PDF page in the existing PDF document using PdfPageRotateAngle(<https://help.syncfusion.com/cr/file-formats/Syncfusion.Pdf.Base~Syncfusion.Pdf.PdfPageRotateAngle.html>) as shown in the following code snippet.

C#

```

//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Gets the page.
PdfPageBase loadedPage = loadedDocument.Pages[0] as PdfPageBase;
//Set the rotation for loaded page.
loadedPage.Rotation = PdfPageRotateAngle.RotateAngle90;
// Save the Document

```

```
loadedDocument.Save("Output.pdf");
// Close the Document
loadedDocument.Close(true);
```

VB.NET

```
'Load the PDF document.
Dim loadedDocument As New PdfLoadedDocument("input.pdf")
'Gets the page.
Dim page As PdfPageBase = TryCast(loadedDocument.Pages(0), PdfPageBase)
'Set the rotation for loaded page.
page.Rotation = PdfPageRotateAngle.RotateAngle90
'Save the document.
loadedDocument.Save("Output.pdf")
'Closes the document.
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Gets the page
PdfPageBase loadedPage = loadedDocument.Pages[0] as PdfPageBase;
//Set the rotation for loaded page.
loadedPage.Rotation = PdfPageRotateAngle.RotateAngle90;
//Save the document as stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document.
FileStream docStream = new FileStream("input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Gets the page
PdfPageBase loadedPage = loadedDocument.Pages[0] as PdfPageBase;
//Set the rotation for loaded page.
loadedPage.Rotation = PdfPageRotateAngle.RotateAngle90;
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
```

```
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Gets the page
PdfPageBase loadedPage = loadedDocument.Pages[0] as PdfPageBase;
//Set the rotation for loaded page.
loadedPage.Rotation = PdfPageRotateAngle.RotateAngle90;
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
loadedDocument.Save(memoryStream);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}
```

Splitting a PDF file to individual pages

Essential PDF allows to split the pages of an existing PDF document into multiple individual PDF documents using [Split](#) method of [PdfLoadedDocument](#) class. The following code snippet explains the same.

C#

```
//Load document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("OutputE.pdf");
//Sets pattern.
const string destinationFilePattern = "Output" + "{0}.pdf";
//Split the pages into separate documents.
loadedDocument.Split(destinationFilePattern);
```



```
//close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load document.
Dim loadedDocument As New PdfLoadedDocument("OutputE.pdf")
'Sets pattern.
Const destinationFilePattern As String = "Output" + "{0}.pdf"
'Split the pages into separate documents.
loadedDocument.Split(destinationFilePattern)
'close the document
loadedDocument.Close(True)
```

UWP

```
//Due to platform limitations, Essential PDF supports splitting a PDF file
into individual pages only in Windows Forms, WPF, ASP.NET, and ASP.NET MVC
platforms. However this can be achieved by using the following code snippet.
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
for (int i = 0; i < loadedDocument.PageCount; i++)
{
    //Creates a new document
    PdfDocument document = new PdfDocument();
    //Imports the loaded document page index to the current document
    document.ImportPage(loadedDocument, i);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    //Close the document
    document.Close(true);
    //Save the stream as PDF document file in local machine. Refer to the
    PDF/UWP section for respective code samples
    Save(stream, "Output" + i + ".pdf");
}
```

ASP.NET CORE

```
//Due to platform limitations, Essential PDF supports splitting a PDF file
into individual pages only in Windows Forms, WPF, ASP.NET, and ASP.NET MVC
platforms. However this can be achieved by using the following code snippet.
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
for (int i=0;i<loadedDocument.PageCount;i++)
```

```

{
    //Creates a new document
    PdfDocument document = new PdfDocument();
    //Imports the pages from the loaded document
    document.ImportPage(loadedDocument, i);
    //Create a memory stream
    MemoryStream stream = new MemoryStream();
    //Save the document to stream
    document.Save(stream);
    stream.Position = 0;
    //Close the document
    document.Close(true);
    //Create a file stream
    FileStream fileStream = new FileStream("Output" + i + ".pdf",
    FileMode.Create, FileAccess.Write);
    byte[] bytes = stream.ToArray();
    //Write bytes to file
    fileStream.Write(bytes, 0, (int)bytes.Length);
    //Dispose the streams
    stream.Dispose();
    fileStream.Dispose();
}

```

XAMARIN

```

//Due to platform limitations, Essential PDF supports splitting a PDF file
into individual pages only in Windows Forms, WPF, ASP.NET, and ASP.NET MVC
platforms. However this can be achieved by using the following code snippet.
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
for (int i = 0; i < loadedDocument.Pages.Count; i++)
{
    //Creates a new document
    PdfDocument document = new PdfDocument();
    //Imports the pages from the loaded document
    document.ImportPage(loadedDocument, i);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    //Close the document
    document.Close(true);
    //Save the stream into PDF file
    //The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
    if (Device.RuntimePlatform == Device.UWP)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output"+ i+
        ".pdf", "application/pdf", stream);
    }
    else
    {

```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output" + i + ".pdf",
"application/pdf", stream);
}
}
```

Working with Text

Drawing text in a new document

You can add text in the new PDF document by using [DrawString](#) method of [PdfGraphics](#) class as shown in the following code sample.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Set the standard font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
'Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0,
0))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
```

```
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
```

```
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Drawing text in an existing document

The following code snippet illustrates how to add text in the existing PDF document by using [DrawString](#) method.

C#

```
//Load a PDF document.
PdfLoadedDocument doc = new PdfLoadedDocument("input.pdf");
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document.
doc.Save("Output.pdf");
//Close the document.
doc.Close(true);
```

VB.NET

```
'Load a PDF document.
Dim doc As New PdfLoadedDocument("input.pdf")
'Get first page from document
Dim page As PdfLoadedPage = TryCast(doc.Pages(0), PdfLoadedPage)
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Set the standard font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
'Draw the text.
```

```
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0))
'Save the document.
doc.Save("Output.pdf")
'Close the document.
doc.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument doc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await doc.OpenAsync(file);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open, FileAccess.Read);
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
doc.Save(stream);
```

```
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf ");
//Load a PDF document.
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Drawing text using different fonts

Essential PDF allows you to add text to the PDF document using the following types of fonts.

1. Standard fonts

2. TrueType fonts
3. Chinese, Japanese and Korean (CJK) fonts

Draw text using standard fonts

PDF has fourteen base fonts, also known as standard fonts which has special significance. The details can be referred from the link below.

Standard type 1 fonts

You can add text using the standard PDF fonts, by initializing [PdfFont](#) class as [PdfStandardFont](#) class. The following code snippet illustrates this.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Set the standard font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
'Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0, 0))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
```



```
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
```

```
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Draw text using TrueType fonts

You can add text using the TrueType fonts installed in the system, by initializing [PdfFont](#) class as [PdfTrueTypeFont](#) class. The following code snippet illustrates this.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Use the font installed in the machine
PdfFont font = new PdfTrueTypeFont(new Font("Arial", 14));
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Use the font installed in the machine
Dim font As PdfFont = New PdfTrueTypeFont(New Font("Arial", 14))
'Draw the text.
```

```
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0, 0))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)
```

You can add text using the font file from local file system by providing the path of the TrueType font location. The following code snippet explains the same.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Provide the path of the local *.ttf file
PdfFont font = new PdfTrueTypeFont(new Font("Arial.ttf", 14));
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
' Provide the path of the local *.ttf file
Dim font As PdfFont = New PdfTrueTypeFont(New Font("Arial.ttf", 14))
'Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, New PointF(0, 0))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local *.ttf file.
```

```

Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Arial.ttf");
//Initialize the PDF TrueType font.
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local *.ttf file.
FileStream fontStream = new FileStream("Arial.ttf", FileMode.Open,
FileAccess.Read);
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;

```

```

//Load the TrueType font.
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Arial.ttf");
//Initialize the PDF TrueType font.
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text.
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Draw text using CJK fonts

You can add text using CJK fonts, initializing [PdfFont](#) class as [PdfCjkStandardFont](#) class. The following code sample illustrates this.

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfCjkStandardFont(PdfCjkFontFamily.HeiseiMinchoW3, 20);
//Draw the text.
graphics.DrawString("こんにちは世界", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);

```

VB.NET

```
'Create a new PDF document.
```

```

Dim document As New PdfDocument()
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Set the standard font.
Dim font As PdfFont = New
PdfCjkStandardFont(PdfCjkFontFamily.HeiseiMinchoW3, 20)
'Draw the text.
graphics.DrawString("こんにちは世界", font, PdfBrushes.Black, New PointF(0,
0))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfCjkStandardFont(PdfCjkFontFamily.HeiseiMinchoW3, 20);
//Draw the text.
graphics.DrawString("こんにちは世界", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfCjkStandardFont(PdfCjkFontFamily.HeiseiMinchoW3, 20);
//Draw the text.
graphics.DrawString("こんにちは世界", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);

```

```
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the standard font.
PdfFont font = new PdfCjkStandardFont(PdfCjkFontFamily.HeiseiMinchoW3, 20);
//Draw the text.
graphics.DrawString("こんにちは世界", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Measuring a string

The Essential PDF allows you to measure the size of a string which uses the PdfFont through [MeasureString](#) method of it and returns the size. Refer to the following code sample.

C#

```
//Create the new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
```

```

PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a new PDF font instance
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12);
string text = "Hello World!";
//Measure the text
SizeF size = font.MeasureString(text);
//Draw string to the PDF page
graphics.DrawString(text, font, PdfBrushes.Black, new
RectangleF(PointF.Empty, size));
//Save the document
document.Save("Output.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Create the new PDF document
Dim document As New PdfDocument()
'Add a page to the document
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create a new PDF font instance
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 12)
Dim text As String = "Hello World!"
'Measure the text
Dim size As SizeF = font.MeasureString(text)
'Draw string to the PDF page
graphics.DrawString(text, font, PdfBrushes.Black, New
RectangleF(PointF.Empty, size))
'Save the document
document.Save("Output.pdf")
'Close the document
document.Close(True)

```

UWP

```

//Create the new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a new PDF font instance
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12);
string text = "Hello World!";
//Measure the text
SizeF size = font.MeasureString(text);
//Draw string to the PDF page
graphics.DrawString(text, font, PdfBrushes.Black, new
RectangleF(PointF.Empty, size));
//Save the document as stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);

```



```
//Close the document instances
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create the new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a new PDF font instance
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12);
string text = "Hello World!";
//Measure the text
SizeF size = font.MeasureString(text);
//Draw string to the PDF page
graphics.DrawString(text, font, PdfBrushes.Black, new
RectangleF(PointF.Empty, size));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty.
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file.
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create the new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a new PDF font instance
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12);
string text = "Hello World!";
//Measure the text
SizeF size = font.MeasureString(text);
//Draw string to the PDF page
graphics.DrawString(text, font, PdfBrushes.Black, new
RectangleF(PointF.Empty, size));
//Save the document as stream
MemoryStream stream = new MemoryStream();
```

```
document.Save(stream);
//Close the document instances
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Embedding fonts and working with Unicode text

To embed a font or display Unicode text in the document, the 'Unicode' Boolean parameter of the [PdfTrueTypeFont](#) constructor has to be set to true. The following code illustrates the same.

Note: To render a Unicode text in the PDF document the chosen font should have the Unicode rendering capability.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//use the system installed font
PdfFont font = new PdfTrueTypeFont(new Font("Arial Unicode MS", 14), true);
//Read the unicode text from the text file.
StreamReader reader = new StreamReader(@"input.txt", Encoding.Unicode);
string text = reader.ReadToEnd();
reader.Close();
//Draw the text.
graphics.DrawString(text, font, PdfBrushes.Black, new PointF(0, 0));
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
```

```

'Use the system installed font
Dim font As PdfFont = New PdfTrueTypeFont(New Font("Arial Unicode MS", 14),
True)
'Read the unicode text from the text file.
Dim reader As New StreamReader("input.txt", Encoding.Unicode)
Dim text As String = reader.ReadToEnd()
reader.Close()
'Draw the text.
graphics.DrawString(text, font, PdfBrushes.Black, New PointF(0, 0))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)

```

UWP

```

//PDF supports embedding fonts or displaying a Unicode text in the PDF
document by default in UWP platform.

```

XAMARIN

```

//PDF supports embedding fonts or displaying a Unicode text in the PDF
document by default in Xamarin platform.

```

ASP.NET CORE

```

//PDF supports embedding fonts or displaying a Unicode text in the PDF
document by default in ASP.NET Core platform.

```

Drawing Right-To-Left text

The Essential PDF allows you to draw the right-to-left language text in a PDF document. To draw RTL scripts such as Arabic, Hebrew, Persian, and Urdu, set the value of [TextDirection](#) property in the [PdfStringFormat](#) class to [RightToLeft](#) using [PdfTextDirection](#) Enum. The languages (e.g., Sindhi and Kurdish) that have more than one script and can be written in either right-to-left or left-to-right format. The [LeftToRight](#) value of the [TextDirection](#) property is used to draw RTL text in the left-to-right format. Refer to the following code sample.

C#

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create font
PdfFont font = new PdfTrueTypeFont(new Font("Arial", 14), true);
//Set the format for string
PdfStringFormat format = new PdfStringFormat();
//Set right-to-left text direction for RTL text
format.TextDirection = PdfTextDirection.RightToLeft;
//Set the text alignment
format.Alignment = PdfTextAlignment.Right;
format.ParagraphIndent = 35f;

```

```

//Read the text from file
StreamReader reader = new StreamReader("Arabic.txt", Encoding.Unicode);
string text = reader.ReadToEnd();
reader.Close();
//Draw string with right-to-left format
graphics.DrawString(text, font, PdfBrushes.Black, new RectangleF(0, 0,
page.GetClientSize().Width, page.GetClientSize().Height), format);
//Set left-to-right text direction for RTL text
format.TextDirection = PdfTextDirection.LeftToRight;
//Set the text alignment
format.Alignment = PdfTextAlignment.Left;
//Draw string with left-to-right format
graphics.DrawString(text, font, PdfBrushes.Black, new RectangleF(0, 100,
page.GetClientSize().Width, page.GetClientSize().Height), format);
//Save the document
doc.Save("Output.pdf");
//Close the document
doc.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim doc As PdfDocument = New PdfDocument()
'Add a page to the document
Dim page As PdfPage = doc.Pages.Add()
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create font
Dim font As PdfFont = New PdfTrueTypeFont(New Font("Arial", 14), True)
'Set the format for string
Dim format As PdfStringFormat = New PdfStringFormat()
'Set right-to-left text direction for RTL text
format.TextDirection = PdfTextDirection.RightToLeft
'Set the alignment
format.Alignment = PdfTextAlignment.Right
format.ParagraphIndent = 35.0F
'Read the text from file
Dim reader As StreamReader = New StreamReader("Arabic.txt",
Encoding.Unicode)
Dim text As String = reader.ReadToEnd()
reader.Close()
'Draw string with right-to-left format
graphics.DrawString(text, font, PdfBrushes.Black, new RectangleF(0, 0,
page.GetClientSize().Width, page.GetClientSize().Height), format)
'Set left-to-right text direction for RTL text
format.TextDirection = PdfTextDirection.LeftToRight
'Set the text alignment
format.Alignment = PdfTextAlignment.Left
'Draw string with left-to-right format
graphics.DrawString(text, font, PdfBrushes.Black, new RectangleF(0, 100,
page.GetClientSize().Width, page.GetClientSize().Height), format)
'Save the document
doc.Save("Output.pdf")
'Close the document
doc.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14, PdfFontStyle.Regular);
//Set the format for string
PdfStringFormat format = new PdfStringFormat();
//Set right-to-left text direction for RTL text
format.TextDirection = PdfTextDirection.RightToLeft;
//Set the alignment
format.Alignment = PdfTextAlignment.Right;
format.ParagraphIndent = 35f;
//Read the text from file
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Arabic.txt");
StreamReader reader = new StreamReader(inputStream);
string text = reader.ReadToEnd();
reader.Dispose();
//Draw string with right-to-left format
graphics.DrawString(text, font, PdfBrushes.Black, new RectangleF(0, 0,
page.ClientSize().Width, page.ClientSize().Height), format);
//Set left-to-right text direction for RTL text
format.TextDirection = PdfTextDirection.LeftToRight;
//Set the text alignment
format.Alignment = PdfTextAlignment.Left;
//Draw string with left-to-right format
graphics.DrawString(text, font, PdfBrushes.Black, new RectangleF(0, 100,
page.ClientSize().Width, page.ClientSize().Height), format);
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create font

```

```

FileStream fontStream = new FileStream("arial.ttf", FileMode.Open,
    FileAccess.Read);
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Set the format for string
PdfStringFormat format = new PdfStringFormat();
//Set right-to-left text direction for RTL text
format.TextDirection = PdfTextDirection.RightToLeft;
//Set the alignment
format.Alignment = PdfTextAlignment.Right;
format.ParagraphIndent = 35f;
//Read the text from file
FileStream rtlText = new FileStream("Arabic.txt", FileMode.Open,
    FileAccess.Read);
StreamReader reader = new StreamReader(rtlText, Encoding.Unicode);
string text = reader.ReadToEnd();
reader.Dispose();
//Draw string with right-to-left format
graphics.DrawString(text, font, PdfBrushes.Black, new RectangleF(0, 0,
    page.GetClientSize().Width, page.GetClientSize().Height), format);
//Set left-to-right text direction for RTL text
format.TextDirection = PdfTextDirection.LeftToRight;
//Set the text alignment
format.Alignment = PdfTextAlignment.Left;
//Draw string with left-to-right format
graphics.DrawString(text, font, PdfBrushes.Black, new RectangleF(0, 100,
    page.GetClientSize().Width, page.GetClientSize().Height), format);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
doc.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
doc.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font
Stream fontStream =
    typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
    sets.arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14, PdfFontStyle.Regular);

```

```

//Read the text from file
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Arabic.txt");
StreamReader reader = new StreamReader(inputStream);
string text = reader.ReadToEnd();
reader.Dispose();
//Set the format for string
PdfStringFormat format = new PdfStringFormat();
//Set the property for RTL text
format.TextDirection = PdfTextDirection.RightToLeft;
//Set the alignment
format.Alignment = PdfTextAlignment.Right;
format.ParagraphIndent = 35f;
//Draw string with right-to-left format
graphics.DrawString(text, font, PdfBrushes.Black, new
Syncfusion.Drawing.RectangleF(0, 0, page.ClientSize().Width,
page.ClientSize().Height), format);
//Set left-to-right text direction for RTL text
format.TextDirection = PdfTextDirection.LeftToRight;
//Set the text alignment
format.Alignment = PdfTextAlignment.Left;
//Draw string with left-to-right format
graphics.DrawString(text, font, PdfBrushes.Black, new
Syncfusion.Drawing.RectangleF(0, 100, page.ClientSize().Width,
page.ClientSize().Height), format);
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Adding a HTML Styled Text

Essential PDF provides support to render simple HTML string in a PDF document that can flow through multiple pages. This can be done by using the [PdfHTMLTextElement](#) class.

1. The PdfHTMLTextElement class provides support for a basic set of HTML tags, to render HTML format text in the PDF document.

Supported tags (Should be XHTML-compliant)

- Font
 - B
 - I
 - U
 - Sub
 - Sup
 - BR
2. The [PdfMetafileLayoutFormat](#) class enables to break the HTML text into multiple pages.
 3. Complex HTML with CSS are not supported in this class. Please use [HTML to PDF](#) section for complex HTML with CSS and URL's

The following code example illustrates how to render the HTML string in a PDF document.

C#

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page to the document.
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Courier, 14);
//Simple HTML content
string htmlText = "<font color='#0000F8'>Essential PDF</font> is a
<u><i>.NET</i></u> " +
"library with the capability to produce Adobe PDF files ";
//Render HtmlText.
PdfHTMLTextElement richTextElement = new PdfHTMLTextElement(htmlText, font,
PdfBrushes.Black);
richTextElement.TextAlign = TextAlign.Left;
//Format Layout.
PdfMetafileLayoutFormat format = new PdfMetafileLayoutFormat();
format.Layout = PdfLayoutType.Paginate;
format.Break = PdfLayoutBreakType.FitPage;
//Draw htmlString.
richTextElement.Draw(page, new RectangleF(0, 20, page.GetClientSize().Width,
page.GetClientSize().Height), format);
//Save the document.
doc.Save("Output.pdf");
//Close the document.
doc.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page to the document.
Dim page As PdfPage = doc.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Set the font.
```



```

Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Courier, 14)
'Simple HTML content
Dim htmlText As String = "<font color='#0000F8'>Essential PDF</font> is a
<u><i>.NET</i></u> " + "library with the capability to produce Adobe PDF
files "
'Render HtmlText.
Dim richTextElement As New PdfHTMLTextElement(htmlText, font,
PdfBrushes.Black)
richTextElement.TextAlign = TextAlign.Left
'Format Layout.
Dim format As New PdfMetafileLayoutFormat()
format.Layout = PdfLayoutType.Paginate
format.Break = PdfLayoutBreakType.FitPage
'Draw htmlString.
richTextElement.Draw(page, New RectangleF(0, 20, page.ClientSize().Width,
page.ClientSize().Height), format)
'Save the document.
doc.Save("Output.pdf")
'Close the document.
doc.Close(True)

```

UWP

```

//PDF supports Adding HTML Styled Text only in Windows Forms, WPF, ASP.NET
and ASP.NET MVC platforms.

```

ASP.NET CORE

```

//PDF supports Adding HTML Styled Text only in Windows Forms, WPF, ASP.NET
and ASP.NET MVC platforms.

```

XAMARIN

```

//PDF supports Adding HTML Styled Text only in Windows Forms, WPF, ASP.NET
and ASP.NET MVC platforms.

```

Creating a multicolumn PDF document

Essential PDF allows you to create multi-column text in PDF document by using [PdfTextElement](#) class.

The following code example illustrates the same.

C#

```

//Create a PDF document instance
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
string text = "Adventure Works Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Create a text element with the text and font

```

```

PdfTextElement textElement = new PdfTextElement(text, new
PdfStandardFont(PdfFontFamily.TimesRoman, 14));
//Draw the text in the first column
textElement.Draw(page, new RectangleF(0, 0, page.GetClientSize().Width / 2,
page.GetClientSize().Height));
textElement = new PdfTextElement(text, new
PdfStandardFont(PdfFontFamily.TimesRoman, 14));
//Draw the text in the second column
textElement.Draw(page, new RectangleF(page.GetClientSize().Width / 2, 0,
page.GetClientSize().Width / 2, page.GetClientSize().Height));
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a PDF document instance
Dim document As New PdfDocument()
'Add page to the document
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
Dim text As String = "Adventure Works Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company. The company manufactures and sells
metal and composite bicycles to North American, European and Asian
commercial markets. While its base operation is located in Washington with
290 employees, several regional sales teams are located throughout their
market base."
'Create a text element with the text and font
Dim textElement As New PdfTextElement(text, New
PdfStandardFont(PdfFontFamily.TimesRoman, 14))
'Draw the text in the first column
textElement.Draw(page, New RectangleF(0, 0, page.GetClientSize().Width / 2,
page.GetClientSize().Height))
textElement = New PdfTextElement(text, New
PdfStandardFont(PdfFontFamily.TimesRoman, 14))
'Draw the text in the second column
textElement.Draw(page, New RectangleF(page.GetClientSize().Width / 2, 0,
page.GetClientSize().Width / 2, page.GetClientSize().Height))
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a PDF document instance
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
string text = "Adventure Works Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Create a text element with the text and font

```

```

PdfTextElement textElement = new PdfTextElement(text, new
PdfStandardFont(PdfFontFamily.TimesRoman, 14));
//Draw the text in the first column
textElement.Draw(page, new RectangleF(0, 0, page.GetClientSize().Width / 2,
page.GetClientSize().Height));
textElement = new PdfTextElement(text, new
PdfStandardFont(PdfFontFamily.TimesRoman, 14));
//Draw the text in the second column
textElement.Draw(page, new RectangleF(page.GetClientSize().Width / 2, 0,
page.GetClientSize().Width / 2, page.GetClientSize().Height));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a PDF document instance
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
string text = "Adventure Works Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Create a text element with the text and font
PdfTextElement textElement = new PdfTextElement(text, new
PdfStandardFont(PdfFontFamily.TimesRoman, 14));
//Draw the text in the first column
textElement.Draw(page, new RectangleF(0, 0, page.GetClientSize().Width / 2,
page.GetClientSize().Height));
textElement = new PdfTextElement(text, new
PdfStandardFont(PdfFontFamily.TimesRoman, 14));
//Draw the text in the second column
textElement.Draw(page, new RectangleF(page.GetClientSize().Width / 2, 0,
page.GetClientSize().Width / 2, page.GetClientSize().Height));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";

```

```
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a PDF document instance
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
string text = "Adventure Works Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Create a text element with the text and font
PdfTextElement textElement = new PdfTextElement(text, new
PdfStandardFont(PdfFontFamily.TimesRoman, 14));
//Draw the text in the first column
textElement.Draw(page, new Syncfusion.Drawing.RectangleF(0, 0,
page.GetClientSize().Width / 2, page.GetClientSize().Height));
textElement = new PdfTextElement(text, new
PdfStandardFont(PdfFontFamily.TimesRoman, 14));
//Draw the text in the second column
textElement.Draw(page, new
Syncfusion.Drawing.RectangleF(page.GetClientSize().Width / 2, 0,
page.GetClientSize().Width / 2, page.GetClientSize().Height));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

The [PdfLayoutFormat](#) class helps to allow the text to flow across pages. The [PdfLayoutResult](#) class provides the rendered bounds of the previously added text which can be used to place successive elements without overlapping.

The following code snippet illustrates how to add elements relatively and also allow the text to flow across multiple pages.

C#

```
//Create a PDF document instance
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Read the long text from the text file.
StreamReader reader = new StreamReader(@"input.txt", Encoding.ASCII);
string text = reader.ReadToEnd();
reader.Close();
const int paragraphGap = 10;
//Create a text element with the text and font
PdfTextElement textElement = new PdfTextElement(text, new
PdfStandardFont(PdfFontFamily.TimesRoman, 14));
PdfLayoutFormat layoutFormat = new PdfLayoutFormat();
layoutFormat.Layout = PdfLayoutType.Paginate;
layoutFormat.Break = PdfLayoutBreakType.FitPage;
//Draw the first paragraph
PdfLayoutResult result = textElement.Draw(page, new RectangleF(0, 0,
page.GetClientSize().Width / 2, page.GetClientSize().Height), layoutFormat);
//Draw the second paragraph from the first paragraph end position
result = textElement.Draw(page, new RectangleF(0, result.Bounds.Bottom +
paragraphGap, page.GetClientSize().Width / 2, page.GetClientSize().Height),
layoutFormat);
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a PDF document instance
Dim document As New PdfDocument()
'Add page to the document
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
'Read the RTL text from the text file.
Dim reader As New StreamReader("input.txt", Encoding.ASCII)
Dim text As String = reader.ReadToEnd()
reader.Close()
Const paragraphGap As Integer = 10
'Create a text element with the text and font
Dim textElement As New PdfTextElement(text, New
PdfStandardFont(PdfFontFamily.TimesRoman, 14))
Dim layoutFormat As New PdfLayoutFormat()
layoutFormat.Layout = PdfLayoutType.Paginate
layoutFormat.Break = PdfLayoutBreakType.FitPage
'Draw the first paragraph
Dim result As PdfLayoutResult = textElement.Draw(page, New RectangleF(0, 0,
page.GetClientSize().Width / 2, page.GetClientSize().Height), layoutFormat)
'Draw the second paragraph from the first paragraph end position
result = textElement.Draw(page, New RectangleF(0, result.Bounds.Bottom +
paragraphGap, page.GetClientSize().Width / 2, page.GetClientSize().Height),
layoutFormat)
```

```
document.Save("Output.pdf");
document.Close(True)
```

UWP

```
//Create a PDF document instance
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Read the long text from the text file.
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.txt");
StreamReader reader = new StreamReader(inputStream, Encoding.ASCII);
string text = reader.ReadToEnd();
reader.Dispose();
const int paragraphGap = 10;
//Create a text element with the text and font
PdfTextElement textElement = new PdfTextElement(text, new
PdfStandardFont(PdfFontFamily.TimesRoman, 14));
PdfLayoutFormat layoutFormat = new PdfLayoutFormat();
layoutFormat.Layout = PdfLayoutType.Paginate;
layoutFormat.Break = PdfLayoutBreakType.FitPage;
//Draw the first paragraph
PdfLayoutResult result = textElement.Draw(page, new RectangleF(0, 0,
page.GetClientSize().Width / 2, page.GetClientSize().Height), layoutFormat);
//Draw the second paragraph from the first paragraph end position
result = textElement.Draw(page, new RectangleF(0, result.Bounds.Bottom +
paragraphGap, page.GetClientSize().Width / 2, page.GetClientSize().Height),
layoutFormat);
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a PDF document instance
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Read the long text from the text file.
FileStream inputStream = new FileStream("Input.txt", FileMode.Open,
FileAccess.Read);
StreamReader reader = new StreamReader(inputStream, Encoding.ASCII);
string text = reader.ReadToEnd();
reader.Dispose();
const int paragraphGap = 10;
//Create a text element with the text and font
```

```

PdfTextElement textElement = new PdfTextElement(text, new
PdfStandardFont(PdfFontFamily.TimesRoman, 14));
PdfLayoutFormat layoutFormat = new PdfLayoutFormat();
layoutFormat.Layout = PdfLayoutType.Paginate;
layoutFormat.Break = PdfLayoutBreakType.FitPage;
//Draw the first paragraph
PdfLayoutResult result = textElement.Draw(page, new RectangleF(0, 0,
page.GetClientSize().Width / 2, page.GetClientSize().Height), layoutFormat);
//Draw the second paragraph from the first paragraph end position
result = textElement.Draw(page, new RectangleF(0, result.Bounds.Bottom +
paragraphGap, page.GetClientSize().Width / 2, page.GetClientSize().Height),
layoutFormat);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a PDF document instance
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Read the long text from the text file.
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.txt");
StreamReader reader = new StreamReader(inputStream, Encoding.UTF8);
string text = reader.ReadToEnd();
reader.Dispose();
const int paragraphGap = 10;
//Create a text element with the text and font
PdfTextElement textElement = new PdfTextElement(text, new
PdfStandardFont(PdfFontFamily.TimesRoman, 14));
PdfLayoutFormat layoutFormat = new PdfLayoutFormat();
layoutFormat.Layout = PdfLayoutType.Paginate;
layoutFormat.Break = PdfLayoutBreakType.FitPage;
//Draw the first paragraph
PdfLayoutResult result = textElement.Draw(page, new
Syncfusion.Drawing.RectangleF(0, 0, page.GetClientSize().Width / 2,
page.GetClientSize().Height), layoutFormat);
//Draw the second paragraph from the first paragraph end position

```

```

result = textElement.Draw(page, new Syncfusion.Drawing.RectangleF(0,
result.Bounds.Bottom + paragraphGap, page.GetClientSize().Width / 2,
page.GetClientSize().Height), layoutFormat);
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Inserting Rich Text Format contents

Essential PDF allows you to insert a RTF text into a PDF document by converting it as bitmap or metafile image and rendering it using [FromRtf](#) method of [PdfImage](#) class.

The following code example illustrates how to insert RTF text in PDF document.

C#

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
SizeF bounds = page.GetClientSize();
//Read RTF document.
StreamReader reader = new StreamReader(@"input.rtf", Encoding.ASCII);
string text = reader.ReadToEnd();
reader.Close();
//Convert it to Metafile.
PdfMetafile imageMetafile = (PdfMetafile)PdfImage.FromRtf(text,
bounds.Width, PdfImageType.Metafile);
PdfMetafileLayoutFormat format = new PdfMetafileLayoutFormat();
//Allow text to flow multiple pages without any break.
format.SplitTextLines = true;
//Draws image.
imageMetafile.Draw(page, 0, 0, format);
//Save the document.
doc.Save("Output.pdf");
doc.Close(true);

```

VB.NET

```

'Create a new PDF document.

```



```

Dim doc As New PdfDocument()
'Add a page.
Dim page As PdfPage = doc.Pages.Add()
Dim bounds As.SizeF = page.GetClientSize()
'Read RTF document.
Dim reader As New StreamReader("input.rtf", Encoding.ASCII)
Dim text As String = reader.ReadToEnd()
reader.Close()
'Convert it to Metafile.
Dim imageMetafile As PdfMetafile = DirectCast(PdfImage.FromRtf(text,
bounds.Width, PdfImageType.Metafile), PdfMetafile)
Dim format As New PdfMetafileLayoutFormat()
'Allow text to flow multiple pages without any break.
format.SplitTextLines = True
'Draws image.
imageMetafile.Draw(page, 0, 0, format)
'Save the document.
doc.Save("Output.pdf")
doc.Close(True)

```

UWP

```

//PDF supports Inserting Rich Text Format contents only in Windows Forms,
WPF, ASP.NET and ASP.NET MVC platforms.

```

ASP.NET CORE

```

//PDF supports Inserting Rich Text Format contents only in Windows Forms,
WPF, ASP.NET and ASP.NET MVC platforms.

```

XAMARIN

```

//PDF supports Inserting Rich Text Format contents only in Windows Forms,
WPF, ASP.NET and ASP.NET MVC platforms.

```

Note: For converting complex RTF content to PDF, refer the [RTF to PDF](#) section.

Adding an Ordered List

Essential PDF allows you to create an ordered list in the document. Ordered List is represented by the [PdfOrderedList](#) class and can be numerical or alphabetical. The following code snippet illustrates the same.

C#

```

//Create a new instance of PdfDocument class.
PdfDocument document = new PdfDocument();
//Add a new page to the document.
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
.SizeF size = page.Graphics.ClientSize;
//Create font
PdfFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 10,
PdfFontStyle.Italic);

```

```

string[] products = { "Tools", "Grid", "Chart", "Edit", "Diagram", "XlsIO",
    "Grouping", "Calculate", "PDF", "HTMLUI", "DocIO" };
//Create string format
PdfStringFormat format = new PdfStringFormat();
format.LineSpacing = 10f;
//Create Ordered list
PdfOrderedList pdfList = new PdfOrderedList();
pdfList.Marker.Brush = PdfBrushes.Black;
pdfList.Indent = 20;
//Set format for sub list
pdfList.Font = font;
pdfList.StringFormat = format;
foreach (string s in products)
{
    //Add items
    pdfList.Items.Add(string.Concat("Essential ", s));
}
pdfList.Draw(page, new RectangleF(0, 20, size.Width, size.Height));
// Save and close the document.
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a new instance of PdfDocument class.
Dim document As New PdfDocument()
'Add a new page to the document.
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
Dim size As.SizeF = page.Graphics.ClientSize
'Create font
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.TimesRoman, 10,
    PdfFontStyle.Italic)
Dim products As String() = {"Tools", "Grid", "Chart", "Edit", "Diagram",
    "XlsIO",
    "Grouping", "Calculate", "PDF", "HTMLUI", "DocIO"}
'Create string format
Dim format As New PdfStringFormat()
format.LineSpacing = 10.0F
'Create Ordered list
Dim pdfList As New PdfOrderedList()
pdfList.Marker.Brush = PdfBrushes.Black
pdfList.Indent = 20
'Set format for sub list
pdfList.Font = font
pdfList.StringFormat = format
For Each s As String In products
    'Add items
    pdfList.Items.Add(String.Concat("Essential ", s))
Next
pdfList.Draw(page, New RectangleF(0, 20, size.Width, size.Height))
' Save and close the document.
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new instance of PdfDocument class.
PdfDocument document = new PdfDocument();
//Add a new page to the document.
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
SizeF size = page.Graphics.ClientSize;
//Create font
PdfFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 10,
PdfFontStyle.Italic);
string[] products = { "Tools", "Grid", "Chart", "Edit", "Diagram", "XlsIO",
"Grouping", "Calculate", "PDF", "HTMLUI", "DocIO" };
//Create string format
PdfStringFormat format = new PdfStringFormat();
format.LineSpacing = 10f;
//Create Ordered list
PdfOrderedList pdfList = new PdfOrderedList();
pdfList.Marker.Brush = PdfBrushes.Black;
pdfList.Indent = 20;
//Set format for sub list
pdfList.Font = font;
pdfList.StringFormat = format;
foreach (string s in products)
{
//Add items
pdfList.Items.Add(string.Concat("Essential ", s));
}
pdfList.Draw(page, new RectangleF(0, 20, size.Width, size.Height));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new instance of PdfDocument class.
PdfDocument document = new PdfDocument();
//Add a new page to the document.
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
SizeF size = page.Graphics.ClientSize;
//Create font
PdfFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 10,
PdfFontStyle.Italic);
string[] products = { "Tools", "Grid", "Chart", "Edit", "Diagram", "XlsIO",
"Grouping", "Calculate", "PDF", "HTMLUI", "DocIO" };
//Create string format
PdfStringFormat format = new PdfStringFormat();
format.LineSpacing = 10f;
//Create Ordered list
PdfOrderedList pdfList = new PdfOrderedList();

```

```

pdfList.Marker.Brush = PdfBrushes.Black;
pdfList.Indent = 20;
//Set format for sub list
pdfList.Font = font;
pdfList.StringFormat = format;
foreach (string s in products)
{
    //Add items
    pdfList.Items.Add(string.Concat("Essential ", s));
}
pdfList.Draw(page, new RectangleF(0, 20, size.Width, size.Height));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new instance of PdfDocument class.
PdfDocument document = new PdfDocument();
//Add a new page to the document.
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
Syncfusion.Drawing.SizeF size = page.Graphics.ClientSize;
//Create font
PdfFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 10,
PdfFontStyle.Italic);
string[] products = { "Tools", "Grid", "Chart", "Edit", "Diagram", "XlsIO",
"Grouping", "Calculate", "PDF", "HTMLUI", "DocIO" };
//Create string format
PdfStringFormat format = new PdfStringFormat();
format.LineSpacing = 10f;
//Create Ordered list
PdfOrderedList pdfList = new PdfOrderedList();
pdfList.Marker.Brush = PdfBrushes.Black;
pdfList.Indent = 20;
//Set format for sub list
pdfList.Font = font;
pdfList.StringFormat = format;
foreach (string s in products)
{
    //Add items
    pdfList.Items.Add(string.Concat("Essential ", s));
}

```

```
pdfList.Draw(page, new Syncfusion.Drawing.RectangleF(0, 20, size.Width,
size.Height));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Adding an Unordered List

Essential PDF also provides support to create an unordered List that is represented by the [PdfUnorderedList](#) class. An Unordered list can be bullets, circle or an image. The following code snippet illustrates the same.

C#

```
//Create a new instance of PdfDocument class.
PdfDocument document = new PdfDocument();
//Add a new page to the document.
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
SizeF size = page.Graphics.ClientSize;
//Create an unordered list
PdfUnorderedList list = new PdfUnorderedList();
//Set the marker style
list.Marker.Style = PdfUnorderedMarkerStyle.Disk;
//Create font and write title
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Regular);
//Create string format
PdfStringFormat format = new PdfStringFormat();
format.LineSpacing = 10f;
// Format list
list.Font = font;
list.StringFormat = format;
//Set list indent
list.Indent = 10;
//Add items to the list
list.Items.Add("PDF");
list.Items.Add("XlsIO");
list.Items.Add("DocIO");
list.Items.Add("PPT");
```

```
//Set text indent
list.TextIndent = 10;
//Draw list
list.Draw(page, new RectangleF(0, 10, size.Width, size.Height));
// Save and close the document.
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new instance of PdfDocument class.
Dim document As New PdfDocument()
'Add a new page to the document.
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
Dim size As SizeF = page.Graphics.ClientSize
'Create an unordered list
Dim list As New PdfUnorderedList()
'Set the marker style
list.Marker.Style = PdfUnorderedMarkerStyle.Disk
'Create font and write title
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Regular)
'Create string format
Dim format As New PdfStringFormat()
format.LineSpacing = 10.0F
' Format list
list.Font = font
list.StringFormat = format
'Set list indent
list.Indent = 10
'Add items to the list
list.Items.Add("PDF")
list.Items.Add("XlsIO")
list.Items.Add("DocIO")
list.Items.Add("PPT")
'Set text indent
list.TextIndent = 10
'Draw list
list.Draw(page, New RectangleF(0, 10, size.Width, size.Height))
'Save and close the document.
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a new instance of PdfDocument class.
PdfDocument document = new PdfDocument();
//Add a new page to the document.
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
SizeF size = page.Graphics.ClientSize;
//Create an unordered list
PdfUnorderedList list = new PdfUnorderedList();
//Set the marker style
list.Marker.Style = PdfUnorderedMarkerStyle.Disk;
```

```

//Create font and write title
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Regular);
//Create string format
PdfStringFormat format = new PdfStringFormat();
format.LineSpacing = 10f;
// Format list
list.Font = font;
list.StringFormat = format;
//Set list indent
list.Indent = 10;
//Add items to the list
list.Items.Add("PDF");
list.Items.Add("XlsIO");
list.Items.Add("DocIO");
list.Items.Add("PPT");
//Set text indent
list.TextIndent = 10;
//Draw list
list.Draw(page, new RectangleF(0, 10, size.Width, size.Height));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new instance of PdfDocument class.
PdfDocument document = new PdfDocument();
//Add a new page to the document.
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
SizeF size = page.Graphics.ClientSize;
//Create an unordered list
PdfUnorderedList list = new PdfUnorderedList();
//Set the marker style
list.Marker.Style = PdfUnorderedMarkerStyle.Disk;
//Create font and write title
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Regular);
//Create string format
PdfStringFormat format = new PdfStringFormat();
format.LineSpacing = 10f;
// Format list
list.Font = font;
list.StringFormat = format;
//Set list indent
list.Indent = 10;
//Add items to the list
list.Items.Add("PDF");
list.Items.Add("XlsIO");
list.Items.Add("DocIO");

```

```

list.Items.Add("PPT");
//Set text indent
list.TextIndent = 10;
//Draw list
list.Draw(page, new RectangleF(0, 10, size.Width, size.Height));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new instance of PdfDocument class.
PdfDocument document = new PdfDocument();
//Add a new page to the document.
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
Syncfusion.Drawing.SizeF size = page.Graphics.ClientSize;
//Create an unordered list
PdfUnorderedList list = new PdfUnorderedList();
//Set the marker style
list.Marker.Style = PdfUnorderedMarkerStyle.Disk;
//Create font and write title
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Regular);
//Create string format
PdfStringFormat format = new PdfStringFormat();
format.LineSpacing = 10f;
// Format list
list.Font = font;
list.StringFormat = format;
//Set list indent
list.Indent = 10;
//Add items to the list
list.Items.Add("PDF");
list.Items.Add("XlsIO");
list.Items.Add("DocIO");
list.Items.Add("PPT");
//Set text indent
list.TextIndent = 10;
//Draw list
list.Draw(page, new Syncfusion.Drawing.RectangleF(0, 10, size.Width,
size.Height));
//Save the document into memory stream.
MemoryStream stream = new MemoryStream();

```



```
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Replace Fonts in an existing document

Essential PDF allows you to replace the fonts in an existing PDF document by using the [Replace](#) method. The following code snippet illustrates the same.

C#

```
//Creates a new PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Replace font
loadedDocument.UsedFonts[0].Replace(new
PdfStandardFont(PdfFontFamily.TimesRoman, 12));
//Save the document
loadedDocument.Save("Output.pdf");
loadedDocument.Close(true);
```

VB.NET

```
'Creates a new PDF document.
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Replace font
loadedDocument.UsedFonts(0).Replace(New
PdfStandardFont(PdfFontFamily.TimesRoman, 12))
'Save the document
loadedDocument.Save("Output.pdf")
loadedDocument.Close(True)
```

UWP

```
//PDF supports Replace fonts in an existing document only in Windows Forms,
WPF, ASP.NET and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//PDF supports Replace fonts in an existing document only in Windows Forms,
WPF, ASP.NET and ASP.NET MVC platforms.
```

XAMARIN

//PDF supports Replace fonts in an existing document only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.

Search and get the bounds of a text in a document

You can search for a particular text in a document and get the bounds using [FindText](#) method of [PdfViewerControl](#) class. To include this functionality, you need to add the below mentioned assemblies as reference to the project.

1. Syncfusion.Compression.Base.dll
2. Syncfusion.Pdf.Base.dll
3. Syncfusion.PdfViewer.Windows.dll

The following code snippet illustrates how to get the bound of a text from PDF document.

C#

```
PdfViewerControl documentViewer = new PdfViewerControl();
//Load the PDF document
documentViewer.Load("Input.pdf");
//Get the occurrences of the target text and location.
Dictionary<int, List<RectangleF>> textSearch = new Dictionary<int,
List<RectangleF>>();
bool IsMatchFound = documentViewer.FindText("hello", out textSearch);
documentViewer.Dispose();
```

VB.NET

```
Dim documentViewer As New PdfViewerControl()
'Load the PDF document
documentViewer.Load("Input.pdf")
'Get the occurrences of the target text and location.
Dim textSearch As New Dictionary(Of Integer, List(Of RectangleF))()
Dim IsMatchFound As Boolean = documentViewer.FindText("hello", textSearch)
documentViewer.Dispose()
```

Drawing complex script language text

Essential PDF allows you to add complex script language text in the PDF document by using the [ComplexScript](#) property available in [PdfStringFormat](#) class. The following code snippet illustrates this.

C#

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the font with Unicode option
Font font = new Font("Tahoma", 14);
PdfFont pdfFont = new PdfTrueTypeFont(font, true);
```

```
//Set the format for string
PdfStringFormat format = new PdfStringFormat();
//Set the format as complex script layout type
format.ComplexScript = true;
//Draw the text
graphics.DrawString("สวัสดิ์ชาวโลก", pdfFont, PdfBrushes.Black, new
RectangleF(0, 0, page.ClientSize().Width, page.ClientSize().Height),
format);
//Save the document
doc.Save("Output.pdf");
//Close the document
doc.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim doc As New PdfDocument()
'Add a page to the document
Dim page As PdfPage = doc.Pages.Add()
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Set the font with Unicode option
Dim font As New Font("Tahoma", 14)
Dim pdfFont As PdfFont = New PdfTrueTypeFont(font, True)
'Set the format for string
Dim format As New PdfStringFormat()
'Set the format as complex script layout type
format.ComplexScript = True
'Draw the text
graphics.DrawString("สวัสดิ์ชาวโลก", pdfFont, PdfBrushes.Black, New
RectangleF(0, 0, page.ClientSize().Width, page.ClientSize().Height),
format)
'Save the document
doc.Save("Output.pdf")
'Close the document
doc.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the font with Unicode option
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("ComplexSc
riptSample.Assets.tahoma.ttf");
//Create a new PDF font instance
PdfFont font = new PdfTrueTypeFont(fontStream, 10);
//Set the format for string
PdfStringFormat format = new PdfStringFormat();
//Set the format as complex script layout type
format.ComplexScript = true;
```

```
//Draw the text
graphics.DrawString("สวัสดิ์ชาวโลก", pdfFont, PdfBrushes.Black, new
RectangleF(0, 0, page.ClientSize().Width, page.ClientSize().Height),
format);
//Save the PDF document
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the PDF document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
FileStream fontStream = new FileStream("tahoma.ttf", FileMode.Open,
FileAccess.Read);
//Create a new PDF font instance
PdfFont font = new PdfTrueTypeFont(fontStream, 10);
//Set the format for string
PdfStringFormat format = new PdfStringFormat();
//Set the format as complex script layout type
format.ComplexScript = true;
//Draw the text
graphics.DrawString("สวัสดิ์ชาวโลก", pdfFont, PdfBrushes.Black, new
RectangleF(0, 0, page.ClientSize().Width, page.ClientSize().Height),
format);
//Save the PDF document
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the PDF document
doc.Close(true);
//Defining the content type for PDF file.
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the font as stream
```

```

Stream fontStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
tahoma.ttf");
//Create a new PDF font instance
PdfFont font = new PdfTrueTypeFont(fontStream, 10);
//Set the format for string
PdfStringFormat format = new PdfStringFormat();
//Set the format as complex script layout type
format.ComplexScript = true;
//Draw the text
graphics.DrawString("สวัสดิ์ชาวโลก", pdfFont, PdfBrushes.Black, new
RectangleF(0, 0, page.ClientSize().Width, page.ClientSize().Height),
format);
//Save the PDF document
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the PDF document
doc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

You can add the complex script language text in an existing PDF document by using the following code sample.

C#

```

//Load a PDF document
PdfLoadedDocument doc = new PdfLoadedDocument("input.pdf");
//Get first page from the document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the font with Unicode option
Font font = new Font("Tahoma", 14);
PdfFont pdfFont = new PdfTrueTypeFont(font, true);
//Set the format for string
PdfStringFormat format = new PdfStringFormat();
//Set the format as complex script layout type
format.ComplexScript = true;
//Draw the text
graphics.DrawString("สวัสดิ์ชาวโลก", pdfFont, PdfBrushes.Black, new
RectangleF(0, 0, page.Size.Width, page.Size.Height), format);
//Save the document

```

```
doc.Save("Output.pdf");
//Close the document
doc.Close(true);
```

VB.NET

```
'Load a PDF document
Dim doc As New PdfLoadedDocument("input.pdf")
'Get first page from the document
Dim page As PdfLoadedPage = TryCast(doc.Pages(0), PdfLoadedPage)
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Set the font with Unicode option
Dim font As New Font("Tahoma", 14)
Dim pdfFont As PdfFont = New PdfTrueTypeFont(font, True)
'Set the format for string
Dim format As New PdfStringFormat()
'Set the format as complex script layout type
format.ComplexScript = True
'Draw the text
graphics.DrawString("สวัสดิ์ชาวโลก", pdfFont, PdfBrushes.Black, New
RectangleF(0, 0, page.Size.Width, page.Size.Height), format)
'Save the document
doc.Save("Output.pdf")
'Close the document
doc.Close(True)
```

UWP

```
Stream inputFileStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("input.pdf
");
//Load a PDF document
PdfLoadedDocument doc = new PdfLoadedDocument(inputFileStream);
//Get first page from the document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the font with Unicode option
Font font = new Font("Tahoma", 14);
PdfFont pdfFont = new PdfTrueTypeFont(font, true);
//Set the format for string
PdfStringFormat format = new PdfStringFormat();
//Set the format as complex script layout type
format.ComplexScript = true;
//Draw the text
graphics.DrawString("สวัสดิ์ชาวโลก", pdfFont, PdfBrushes.Black, new
RectangleF(0, 0, page.Size.Width, page.Size.Height), format);
//Save the PDF document
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the PDF document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
```

```
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
FileStream inputFileStream = new FileStream("input.pdf", FileMode.Open,
    FileAccess.Read);
//Load a PDF document
PdfLoadedDocument doc = new PdfLoadedDocument(inputFileStream);
//Get first page from the document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the font with Unicode option
Font font = new Font("Tahoma", 14);
PdfFont pdfFont = new PdfTrueTypeFont(font, true);
//Set the format for string
PdfStringFormat format = new PdfStringFormat();
//Set the format as complex script layout type
format.ComplexScript = true;
//Draw the text
graphics.DrawString("สวัสดิ์ชาวโลก", pdfFont, PdfBrushes.Black, new
    RectangleF(0, 0, page.Size.Width, page.Size.Height), format);
//Save the PDF document
MemoryStream stream = new MemoryStream();
await doc.Save(stream);
//Close the PDF document
doc.Close(true);
//Defining the content type for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
Stream inputFileStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
    input.pdf");
//Load a PDF document
PdfLoadedDocument doc = new PdfLoadedDocument(inputFileStream);
//Get first page from the document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Set the font with Unicode option
Font font = new Font("Tahoma", 14);
PdfFont pdfFont = new PdfTrueTypeFont(font, true);
//Set the format for string
PdfStringFormat format = new PdfStringFormat();
//Set the format as complex script layout type
format.ComplexScript = true;
//Draw the text
```

```

graphics.DrawString("สวัสดีชาวโลก", pdfFont, PdfBrushes.Black, new
RectangleF(0, 0, page.Size.Width, page.Size.Height), format);
//Save the PDF document
MemoryStream stream = new MemoryStream();
await doc.Save(stream);
//Close the PDF document
doc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Drawing text using OpenType font

Essential PDF supports drawing text on a PDF document with OpenType font using [PdfTrueTypeFont](#) class, by providing the path of font file from local file system. The following code snippet illustrates this.

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create font
Stream fontStream = System.IO.File.OpenRead("Font.otf");
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Text to draw
string text = "Syncfusion Essential PDF is a.NET PDF library used to create,
read, and edit PDF files in any application";
//Get page client size
SizeF clipBounds = page.Graphics.ClientSize;
RectangleF rect = new RectangleF(0, 0, clipBounds.Width, clipBounds.Height);
//Draw the text
page.Graphics.DrawString(text, font, PdfBrushes.Blue, rect);
//Save the document
document.Save("Output.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = document.Pages.Add

```



```

'Create font
Dim fontStream As Stream = System.IO.File.OpenRead("Font.otf")
Dim font As PdfFont = New PdfTrueTypeFont(fontStream, 14)
'Text to draw
Dim text As String = "Syncfusion Essential PDF is a.NET PDF library used to
create, read, and edit PDF files in any application"
'Get page client size
Dim clipBounds As SizeF = page.Graphics.ClientSize
Dim rect As RectangleF = New RectangleF(0, 0, clipBounds.Width,
clipBounds.Height)
'Draw the text
page.Graphics.DrawString(text, font, PdfBrushes.Blue, rect)
'Save the document
document.Save("Output.pdf")
'Close the document
document.Close(true)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create font
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Font.otf");
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Text to draw
string text = @"Syncfusion Essential PDF is a.NET PDF library used to
create, read, and edit PDF files in any application";
//Create a brush
PdfBrush brush = new PdfSolidBrush(new PdfColor(0, 0, 0));
//Create a pen
PdfPen pen = new PdfPen(new PdfColor(0, 0, 0));
//Get page client size
SizeF clipBounds = page.Graphics.ClientSize;
RectangleF rect = new RectangleF(0, 0, clipBounds.Width, clipBounds.Height);
//Draw the text
page.Graphics.DrawString(text, font, brush, rect);
//Save the PDF document
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the PDF document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create font

```

```

FileStream fontFileStream = new FileStream("Font.otf", FileMode.Open,
    FileAccess.Read);
PdfFont font = new PdfTrueTypeFont(fontFileStream, 14);
//Text to draw
string text = "Syncfusion Essential PDF is a.NET Core PDF library used to
create, read, and edit PDF files in any application";
//Create a brush
PdfBrush brush = new PdfSolidBrush(new PdfColor(0, 0, 0));
//Create a pen
PdfPen pen = new PdfPen(new PdfColor(0, 0, 0));
//Get page client size
SizeF clipBounds = page.Graphics.ClientSize;
RectangleF rect = new RectangleF(0, 0, clipBounds.Width, clipBounds.Height);
//Draw the text
page.Graphics.DrawString(text, font, brush, rect);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the PDF document
document.Close(true);
//Defining the content type for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create font
Stream fontStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Font.otf");
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Text to draw
string text = @"Syncfusion Essential PDF is a.NET PDF library used to
create, read, and edit PDF files in any application";
//Create a brush
PdfBrush brush = new PdfSolidBrush(new PdfColor(0, 0, 0));
//Create a pen
PdfPen pen = new PdfPen(new PdfColor(0, 0, 0));
//Get page client size
SizeF clipBounds = page.Graphics.ClientSize;
RectangleF rect = new RectangleF(0, 0, clipBounds.Width, clipBounds.Height);
//Draw the text
page.Graphics.DrawString(text, font, brush, rect);
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the PDF document
document.Close(true);
//Save the stream into PDF file

```

```
//The operation in Save under Xamarin varies between Windows Phone, Android, and iOS platforms. Refer to the PDF/Xamarin section for respective code samples  
if (Device.RuntimePlatform == Device.UWP)  
{  
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf", stream);  
}  
else  
{  
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf", stream);  
}
```

Working with Images

Essential PDF supports both raster and vector images.

Images are supported through the [PdfImage](#) class, which is an abstract base class that provides the common functionality for [PdfBitmap](#) and [PdfMetafile](#) classes.

Inserting an image in a new document

The following raster images are supported in Essential PDF.

- BMP
- JPEG
- JPEG with Exif standard
- GIF
- PNG
- TIFF
- ICO and ICON

You can load image streams, files on disk, and use System.Drawing.Bitmap objects to draw the images through the [DrawImage](#) method of the [PdfGraphics](#) class.

The following code snippet shows how to add a file from disk to the PDF document.

C#

```
//Create a new PDF document  
PdfDocument doc = new PdfDocument();  
//Add a page to the document  
PdfPage page = doc.Pages.Add();  
//Create PDF graphics for the page  
PdfGraphics graphics = page.Graphics;  
//Load the image from the disk  
PdfBitmap image = new PdfBitmap("Autumn Leaves.jpg");  
//Draw the image  
graphics.DrawImage(image, 0, 0);  
//Save the document  
doc.Save("Output.pdf");  
//Close the document  
doc.Close(true);
```

VB.NET

```

'Create a new PDF document
Dim doc As New PdfDocument()
'Add a page to the document
Dim page As PdfPage = doc.Pages.Add()
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Load the image from the disk
Dim image As New PdfBitmap("Autumn Leaves.jpg")
'Draw the image
graphics.DrawImage(image, 0, 0)
'Save the document
doc.Save("Output.pdf")
'Close the document
doc.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image as stream from the disk
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Autumn Leaves.jpg");
PdfBitmap image = new PdfBitmap(imageStream);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the document as stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image from the disk
FileStream imageStream = new FileStream("Autumn Leaves.jpg", FileMode.Open,
FileAccess.Read);
PdfBitmap image = new PdfBitmap(imageStream);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Creating the stream object

```

```

MemoryStream stream = new MemoryStream();
//Save the document as stream
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
doc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image as stream
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Autumn Leaves.jpg");
PdfBitmap image = new PdfBitmap(imageStream);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the document as stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Inserting an image in an existing document

You can also add images into an existing PDF document using the below code snippet.

C#

```
//Load a PDF document
PdfLoadedDocument doc = new PdfLoadedDocument("input.pdf");
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image from the disk
PdfBitmap image = new PdfBitmap("Autumn Leaves.jpg");
//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the document
doc.Save("Output.pdf");
//Close the document
doc.Close(true);
```

VB.NET

```
'Load a PDF document
Dim doc As New PdfLoadedDocument("input.pdf")
'Get first page from document
Dim page As PdfLoadedPage = TryCast(doc.Pages(0), PdfLoadedPage)
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Load the image from the disk
Dim image As New PdfBitmap("Autumn Leaves.jpg")
'Draw the image
graphics.DrawImage(image, 0, 0)
'Save the document
doc.Save("Output.pdf")
'Close the document
doc.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument doc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await doc.OpenAsync(file);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image as stream
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Autumn Leaves.jpg");
PdfBitmap image = new PdfBitmap(imageStream);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the document as stream
```

```
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image from the disk
FileStream imageStream = new FileStream("Autumn Leaves.jpg", FileMode.Open,
FileAccess.Read);
PdfBitmap image = new PdfBitmap(imageStream);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
doc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image as stream
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Autumn Leaves.jpg");
PdfBitmap image = new PdfBitmap(imageStream);
```

```

//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the document as stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

To add image from stream, use the below code snippet.

C#

```

//Load a PDF document
PdfLoadedDocument doc = new PdfLoadedDocument("input.pdf");
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image
Stream imageStream = File.OpenRead("Autumn Leaves.jpg");
//Load the image from the stream
PdfBitmap image = new PdfBitmap(imageStream);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the document
doc.Save("Output.pdf");
//Close the document
doc.Close(true);

```

VB.NET

```

'Load a PDF document
Dim doc As New PdfLoadedDocument("input.pdf")
'Get first page from document
Dim page As PdfLoadedPage = TryCast(doc.Pages(0), PdfLoadedPage)
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Load the image
Dim imageStream As Stream = File.OpenRead("Autumn Leaves.jpg")
'Load the image from the stream
Dim image As New PdfBitmap(imageStream)

```



```
'Draw the image
graphics.DrawImage(image, 0, 0)
'Save the document
doc.Save("Output.pdf")
'Close the document
doc.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument doc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await doc.OpenAsync(file);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image as stream
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Autumn Leaves.jpg");
PdfBitmap image = new PdfBitmap(imageStream);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the document as stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image from the disk
FileStream imageStream = new FileStream("Autumn Leaves.jpg", FileMode.Open,
FileAccess.Read);
PdfBitmap image = new PdfBitmap(imageStream);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Creating the stream object
```

```

MemoryStream stream = new MemoryStream();
//Save the document as stream
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
doc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image as stream
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Autumn Leaves.jpg");
PdfBitmap image = new PdfBitmap(imageStream);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the document as stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Inserting a vector image

Essential PDF supports adding Metafile vector image. During the insertion, Metafile graphics will be transformed to native PDF graphics that supports text selection and searching. The following types of Metafiles are supported in Essential PDF.

- EMF only
- EMF plus
- EMF plus dual
- WMF

[PdfMetafile](#) class is used to load EMF images. Additionally the [PdfMetafileLayoutFormat](#) class allows you to prevent text and image split across pages in the PDF document.

The following code illustrate this,

C#

```
//Create a PDF Document
PdfDocument doc = new PdfDocument();
//Add pages to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create the layout format
PdfMetafileLayoutFormat format = new PdfMetafileLayoutFormat();
//Split text and image between pages
format.SplitImages = true;
format.SplitTextLines = true;
//Create a Metafile instance
PdfMetafile metaChart = new PdfMetafile("MetaChart.emf");
//Draw the Metafile in the page
metaChart.Draw(page, PointF.Empty, format);
//Save the document
doc.Save("Output.pdf");
//Close the document
doc.Close(true);
```

VB.NET

```
'Create a PDF Document
Dim doc As New PdfDocument()
'Add pages to the document
Dim page As PdfPage = doc.Pages.Add()
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create the layout format
Dim format As New PdfMetafileLayoutFormat()
'Split text and image between pages
format.SplitImages = True
format.SplitTextLines = True
'Create a Metafile instance
Dim metaChart As New PdfMetafile("MetaChart.emf")
'Draw the Metafile in the page
metaChart.Draw(page, PointF.Empty, format)
'Save the document
```

```
doc.Save("Output.pdf")
'Close the document
doc.Close(True)
```

UWP

```
//PDF supports inserting a vector image only in Windows Forms, WPF, ASP.NET
and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//PDF supports inserting a vector image only in Windows Forms, WPF, ASP.NET
and ASP.NET MVC platforms.
```

XAMARIN

```
//PDF supports inserting a vector image only in Windows Forms, WPF, ASP.NET
and ASP.NET MVC platforms.
```

Working with image masking

Essential PDF supports image masking through the [PdfImageMask](#) class.

The following code illustrate shows how to add a mask to TIFF image.

C#

```
//Create a PDF document
PdfDocument doc = new PdfDocument();
//Add pages to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TIFF image
PdfBitmap image = new PdfBitmap("image.tif");
//Create masking image
PdfImageMask mask = new PdfImageMask(new PdfBitmap("mask.bmp"));
image.Mask = mask;
//Draw the image
graphics.DrawImage(image, 0, 0);
//Saves the document
doc.Save("Output.pdf");
//Close the document
doc.Close(true);
```

VB.NET

```
'Create a PDF document
Dim doc As New PdfDocument()
'Add pages to the document
Dim page As PdfPage = doc.Pages.Add()
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Load the TIFF image
Dim image As New PdfBitmap("image.tif")
```

```

'Create masking image
Dim mask As New PdfImageMask(New PdfBitmap("mask.bmp"))
image.Mask = mask
'Draw the image
graphics.DrawImage(image, 0, 0)
'Saves the document
doc.Save("Output.pdf")
'Close the document
doc.Close(True)

```

UWP

```

//PDF supports image masking only in Windows Forms, WPF, ASP.NET and ASP.NET
MVC platforms

```

ASP.NET CORE

```

//Create a PDF document
PdfDocument doc = new PdfDocument();
//Add pages to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TIFF image
FileStream imageStream = new FileStream("image.tif", FileMode.Open,
FileAccess.Read);
PdfTiffImage image = new PdfTiffImage(imageStream);
//Create masking image
FileStream maskStream = new FileStream("mask.bmp", FileMode.Open,
FileAccess.Read);
PdfImageMask mask = new PdfImageMask(new PdfTiffImage(maskStream));
image.Mask = mask;
//Draw the image
graphics.DrawImage(image, 0, 0);
///Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
doc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//PDF supports image masking only in Windows Forms, WPF, ASP.NET and ASP.NET
MVC platforms

```

Note: 1. Essential PDF supports image masking with [Syncfusion.Pdf.Imaging.Portable](#) assembly reference in ASP.NET Core.

Replacing Images in an existing PDF document

Essential PDF allows you to replace images in an existing document. The [ReplaceImage](#) method of the page collection allows you to replace an image.

C#

```
//Load the PDF document
PdfLoadedDocument doc = new PdfLoadedDocument(@"image.pdf");
//Create an image instance
PdfBitmap image = new PdfBitmap(@"Autumn Leaves.jpg");
//Replace the first image in the page
doc.Pages[0].ReplaceImage(0, image);
//Save the document
doc.Save("Output.pdf");
//Close the document
doc.Close(true);
```

VB.NET

```
'Load the PDF document
Dim doc As New PdfLoadedDocument("image.pdf")
'Create an image instance
Dim image As New PdfBitmap("Autumn Leaves.jpg")
'Replace the first image in the page
doc.Pages(0).ReplaceImage(0, image)
'Save the document
doc.Save("Output.pdf")
'Close the document
doc.Close(True)
```

UWP

//PDF supports replacing image in an existing PDF document only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.

ASP.NET CORE

//PDF supports replacing image in an existing PDF document only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.

XAMARIN

//PDF supports replacing image in an existing PDF document only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.

Image Pagination

You can allow a large image to paginate across multiple pages in the PDF document. This can be done through the [PdfLayoutFormat](#) class as shown below.

C#

```

//Create Document
PdfDocument doc = new PdfDocument();
//Add new page
PdfPage page = doc.Pages.Add();
//Load a bitmap
PdfBitmap image = new PdfBitmap("input.jpg");
//Set layout property to make the element break across the pages
PdfLayoutFormat format = new PdfLayoutFormat();
format.Break = PdfLayoutBreakType.FitPage;
format.Layout = PdfLayoutType.Paginate;
//Draw image
image.Draw(page, 20, 400, format);
//Save the PDF
doc.Save("output.pdf");
doc.Close(true);

```

VB.NET

```

'Create Document
Dim doc As New PdfDocument()
'Add new page
Dim page As PdfPage = doc.Pages.Add()
'Load a bitmap
Dim image As New PdfBitmap("input.jpg")
'Set layout property to make the element break across the pages
Dim format As New PdfLayoutFormat()
format.Break = PdfLayoutBreakType.FitPage
format.Layout = PdfLayoutType.Paginate
'Draw image
image.Draw(page, 20, 400, format)
'Save the PDF
doc.Save("output.pdf")
doc.Close(True)

```

UWP

```

//Create Document
PdfDocument doc = new PdfDocument();
//Add new page
PdfPage page = doc.Pages.Add();
//Load the image as stream
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Data.Assets.Autumn Leaves.jpg");
//Load a bitmap
PdfBitmap image = new PdfBitmap(imageStream);
//Set layout property to make the element break across the pages
PdfLayoutFormat format = new PdfLayoutFormat();
format.Break = PdfLayoutBreakType.FitPage;
format.Layout = PdfLayoutType.Paginate;
//Draw image
image.Draw(page, 20, 400, format);
//Save the document as stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document

```

```
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create Document
PdfDocument doc = new PdfDocument();
//Add new page
PdfPage page = doc.Pages.Add();
//Load a bitmap
FileStream imageStream = new FileStream("Autumn Leaves.jpg", FileMode.Open,
FileAccess.Read);
PdfBitmap image = new PdfBitmap(imageStream);
//Set layout property to make the element break across the pages
PdfLayoutFormat format = new PdfLayoutFormat();
format.Break = PdfLayoutBreakType.FitPage;
format.Layout = PdfLayoutType.Paginate;
//Draw image
image.Draw(page, 20, 400, format);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
doc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create Document
PdfDocument doc = new PdfDocument();
//Add new page
PdfPage page = doc.Pages.Add();
//Load a bitmap
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Data.Autumn Leaves.jpg");
PdfBitmap image = new PdfBitmap(imageStream);
//Set layout property to make the element break across the pages
PdfLayoutFormat format = new PdfLayoutFormat();
format.Break = PdfLayoutBreakType.FitPage;
format.Layout = PdfLayoutType.Paginate;
//Draw image
image.Draw(page, 20, 400, format);
//Save the document as stream
MemoryStream stream = new MemoryStream();
```



```

doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Applying transparency and rotation to the image

You can add transparency and rotation to the image using [SetTransparency](#) and [RotateTransform](#) methods of [PdfGraphics](#) respectively. This is explained in the below code snippet.

C#

```

//Create Document
PdfDocument doc = new PdfDocument();
//Add a new page
PdfPage page = doc.Pages.Add();
//Load a bitmap
PdfBitmap image = new PdfBitmap("input.jpg");
//save the current graphics state
PdfGraphicsState state = page.Graphics.Save();
//Translate the coordinate system to the required position
page.Graphics.TranslateTransform(20, 100);
//Apply transparency
page.Graphics.SetTransparency(0.5f);
//Rotate the coordinate system
page.Graphics.RotateTransform(-45);
//Draw image
image.Draw(page, 0, 0);
//Restore the graphics state
page.Graphics.Restore(state);
//Save the PDF
doc.Save("output.pdf");
doc.Close(true);

```

VB.NET

```

'Create Document
Dim doc As New PdfDocument()
'Add a new page
Dim page As PdfPage = doc.Pages.Add()
'Load a bitmap
Dim image As New PdfBitmap("input.jpg")

```

```

'save the current graphics state
Dim state As PdfGraphicsState = page.Graphics.Save()
'Translate the coordinate system to the required position
page.Graphics.TranslateTransform(20, 100)
'Apply transparency
page.Graphics.SetTransparency(0.5F)
'Rotate the coordinate system
page.Graphics.RotateTransform(-45)
' Draw image
image.Draw(page, 0, 0)
'Restore the graphics state
page.Graphics.Restore(state)
'Save the PDF
doc.Save("output.pdf")
doc.Close(True)

```

UWP

```

//Create Document
PdfDocument doc = new PdfDocument();
//Add new page
PdfPage page = doc.Pages.Add();
//Load the image as stream
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Data.Assets.input.jpg");
//Load a bitmap
PdfBitmap image = new PdfBitmap(imageStream);
//save the current graphics state
PdfGraphicsState state = page.Graphics.Save();
//Translate the coordinate system to the required position
page.Graphics.TranslateTransform(20, 100);
//Apply transparency
page.Graphics.SetTransparency(0.5f);
//Rotate the coordinate system
page.Graphics.RotateTransform(-45);
//Draw image
image.Draw(page, 0, 0);
//Restore the graphics state
page.Graphics.Restore(state);
//Save the document as stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create Document
PdfDocument doc = new PdfDocument();
//Add a new page
PdfPage page = doc.Pages.Add();
//Load a image as stream

```

```

FileStream imageStream = new FileStream("input.jpg", FileMode.Open,
    FileAccess.Read);
//Load a bitmap
PdfBitmap image = new PdfBitmap(imageStream);
//save the current graphics state
PdfGraphicsState state = page.Graphics.Save();
//Translate the coordinate system to the required position
page.Graphics.TranslateTransform(20, 100);
//Apply transparency
page.Graphics.SetTransparency(0.5f);
//Rotate the coordinate system
page.Graphics.RotateTransform(-45);
//Draw image
image.Draw(page, 0, 0);
//Restore the graphics state
page.Graphics.Restore(state);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
doc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create Document
PdfDocument doc = new PdfDocument();
//Add a new page
PdfPage page = doc.Pages.Add();
//Load a bitmap
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.jpg");
PdfBitmap image = new PdfBitmap(imageStream);
//save the current graphics state
PdfGraphicsState state = page.Graphics.Save();
//Translate the coordinate system to the required position
page.Graphics.TranslateTransform(20, 100);
//Apply transparency
page.Graphics.SetTransparency(0.5f);
//Rotate the coordinate system
page.Graphics.RotateTransform(-45);
//Draw image
image.Draw(page, 0, 0);
//Restore the graphics state
page.Graphics.Restore(state);
//Save the document as stream

```

```

MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Converting multi page TIFF to PDF

Multi frame TIFF image can be converted to PDF document. This can be done by accessing each frame of the multi frame TIFF image and rendering it in each page of the PDF document.

The code snippet to illustrate the same is given below.

C#

```

//Create a PDF document
PdfDocument pdfDocument = new PdfDocument();
//Set page margins
pdfDocument.PageSettings.Margins.All = 0;
//Load multi frame TIFF image
PdfBitmap tiffImage = new PdfBitmap("image.tiff");
//Get the frame count
int frameCount = tiffImage.FrameCount;
//Access each frame and draw into the page
for (int i = 0; i < frameCount; i++)
{
PdfPage page = pdfDocument.Pages.Add();
PdfGraphics graphics = page.Graphics;
tiffImage.ActiveFrame = i;
graphics.DrawImage(tiffImage, 0, 0, page.GetClientSize().Width,
page.GetClientSize().Height);
}
//Save and close the document
pdfDocument.Save("Sample.pdf");
pdfDocument.Close(true);

```

VB.NET

```

'Create a PDF document
Dim pdfDocument As New PdfDocument()
'Set page margins
pdfDocument.PageSettings.Margins.All = 0
'Load multi frame TIFF image

```

```

Dim tiffImage As New PdfBitmap("image.tiff")
'Get the frame count
Dim frameCount As Integer = tiffImage.FrameCount
'Access each frame and draw into the page
For i As Integer = 0 To frameCount - 1
Dim page As PdfPage = pdfDocument.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
tiffImage.ActiveFrame = i
graphics.DrawImage(tiffImage, 0, 0, page.GetClientSize().Width,
page.GetClientSize().Height)
Next
'Save and close the document
pdfDocument.Save("Sample.pdf")
pdfDocument.Close(True)

```

UWP

```

//Create a PDF document
PdfDocument pdfDocument = new PdfDocument();
//Set page margins
pdfDocument.PageSettings.Margins.All = 0;
//Load multi frame TIFF image
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.image.tiff");
PdfBitmap tiffImage = new PdfBitmap(imageStream);
//Get the frame count
int frameCount = tiffImage.FrameCount;
//Access each frame and draw into the page
for (int i = 0; i < frameCount; i++)
{
PdfPage page = pdfDocument.Pages.Add();
PdfGraphics graphics = page.Graphics;
tiffImage.ActiveFrame = i;
graphics.DrawImage(tiffImage, 0, 0, page.GetClientSize().Width,
page.GetClientSize().Height);
}
MemoryStream memoryStream = new MemoryStream();
//Save the document
await pdfDocument.SaveAsync(memoryStream);
//Close the documents
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(memoryStream, "Sample.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Set page margins
doc.PageSettings.Margins.All = 0;
//Load the multi frame TIFF image from the disk
FileStream imageStream = new FileStream("image.tiff", FileMode.Open,
FileAccess.Read);
PdfTiffImage tiffImage = new PdfTiffImage(imageStream);

```

```
//Get the frame count
int frameCount = tiffImage.FrameCount;
//Access each frame and draw into the page
for (int i = 0; i < frameCount; i++)
{
    PdfPage page = doc.Pages.Add();
    PdfGraphics graphics = page.Graphics;
    tiffImage.ActiveFrame = i;
    graphics.DrawImage(tiffImage, 0, 0, page.GetClientSize().Width,
        page.GetClientSize().Height);
}
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
doc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Essential PDF supports converting multi page TIFF to PDF only in Windows
Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms.
```

Note: 1. Essential PDF supports converting TIFF to PDF with [Syncfusion.Pdf.Imaging.Portable](#) assembly reference in ASP.NET Core.

Working with Brushes

Brushes are used to draw the content on PDF document with specific color and style. Various brushes available in Syncfusion Essential PDF are,

1. Solid Brush
2. Gradient Brush
 - Linear Gradient Brush
 - Radial Gradient Brush
3. Tiling Brush

Solid Brush

The solid brush is used to fill an object with solid color. Essential PDF supports drawing shapes on PDF document with solid brush using the [PdfSolidBrush](#) class. The following code snippet illustrates this.

C#

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
```

```
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF solid brush
PdfSolidBrush brush = new PdfSolidBrush(Color.Red);
//Draw ellipse on the page
graphics.DrawEllipse(brush, new RectangleF(0, 0, 200, 100));
//Save the PDF document
doc.Save("SolidBrush.pdf");
//Close the instance of PdfDocument
doc.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim doc As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = doc.Pages.Add
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create new PDF solid brush
Dim brush As PdfSolidBrush = New PdfSolidBrush(Color.Red)
'Draw ellipse on the page
graphics.DrawEllipse(brush, New RectangleF(0, 0, 200, 100))
'Save the PDF document
doc.Save("SolidBrush.pdf")
'Close the instance of PdfDocument
doc.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF solid brush
PdfSolidBrush brush = new PdfSolidBrush(Color.FromArgb(255, 255, 0, 0));
//Draw ellipse on the page
graphics.DrawEllipse(brush, new RectangleF(0, 0, 200, 100));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the instance of PdfDocument
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "SolidBrush.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
```

```
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF solid brush
PdfSolidBrush brush = new PdfSolidBrush(Color.Red);
//Draw ellipse on the page
graphics.DrawEllipse(brush, new RectangleF(0, 0, 200, 100));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileName = "SolidBrush.pdf";
return fileStreamResult;
```

XAMARIN

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF solid brush
PdfSolidBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Red);
//Draw ellipse on the page
graphics.DrawEllipse(brush, new RectangleF(0, 0, 200, 100));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the instance of PdfDocument
doc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("SolidBrush.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("SolidBrush.pdf", "application/pdf", stream);
}
```


Linear gradient brush

The gradient brush is used to fill an object with blend of two or more colors. Essential PDF supports drawing shapes on PDF document with linear gradient brush using [PdfLinearGradientBrush](#). The following code snippet illustrates this.

C#

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF linear gradient brush
PdfLinearGradientBrush brush = new PdfLinearGradientBrush(new PointF(0, 0),
new PointF(200, 100), Color.Red, Color.Blue);
//Draw ellipse on the page
graphics.DrawEllipse(brush, new RectangleF(0, 0, 200, 100));
//Save the PDF document
doc.Save("LinearGradientBrush.pdf");
//Close the instance of PdfDocument
doc.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim doc As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = doc.Pages.Add
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create new PDF linear gradient brush
Dim brush As PdfLinearGradientBrush = New PdfLinearGradientBrush(New
PointF(0, 0), New PointF(200, 100), Color.Red, Color.Blue)
'Draw ellipse on the page
graphics.DrawEllipse(brush, New RectangleF(0, 0, 200, 100))
'Save the PDF document
doc.Save("LinearGradientBrush.pdf")
'Close the instance of PdfDocument
doc.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF linear gradient brush
PdfLinearGradientBrush brush = new PdfLinearGradientBrush(new PointF(0, 0),
new PointF(200, 100), Color.FromArgb(255, 255, 0, 0), Color.FromArgb(255, 0,
0, 255));
//Draw ellipse on the page
graphics.DrawEllipse(brush, new RectangleF(0, 0, 200, 100));
```

```
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the instance of PdfDocument
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "LinearGradientBrush.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF linear gradient brush
PdfLinearGradientBrush brush = new PdfLinearGradientBrush(new PointF(0, 0),
new PointF(200, 100), Color.Red, Color.Blue);
//Draw ellipse on the page
graphics.DrawEllipse(brush, new RectangleF(0, 0, 200, 100));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "LinearGradientBrush.pdf";
return fileStreamResult;
```

XAMARIN

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF linear gradient brush
PdfLinearGradientBrush brush = new PdfLinearGradientBrush(new PointF(0, 0),
new PointF(200, 100), Syncfusion.Drawing.Color.Red,
Syncfusion.Drawing.Color.Blue);
//Draw ellipse on the page
graphics.DrawEllipse(brush, new RectangleF(0, 0, 200, 100));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the instance of PdfDocument
doc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("LinearGradientBrush.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("LinearGradientBrush.pdf",
"application/pdf", stream);
}

```

Radial Gradient Brush

The gradient brush is used to fill an object with blend of two or more colors. You can draw any shape on PDF document with radial gradient brush using [PdfRadialGradientBrush](#). The following code snippet explains this.

C#

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF radial gradient brush
PdfRadialGradientBrush brush = new PdfRadialGradientBrush(new PointF(50,
50), 0, new PointF(50, 50), 50, Color.Red, Color.Blue);
//Draw ellipse on the page
graphics.DrawEllipse(brush, new RectangleF(0, 0, 100, 100));
//Save the PDF document
doc.Save("RadialGradientBrush.pdf");
//Close the instance of PdfDocument
doc.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim doc As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = doc.Pages.Add
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create new PDF radial gradient brush
Dim brush As PdfRadialGradientBrush = New PdfRadialGradientBrush(New
PointF(50, 50), 0, New PointF(50, 50), 50, Color.Red, Color.Blue)
'Draw ellipse on the page
graphics.DrawEllipse(brush, New RectangleF(0, 0, 100, 100))
'Save the PDF document
doc.Save("RadialGradientBrush.pdf")
'Close the instance of PdfDocument
doc.Close(True)

```

UWP

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF radial gradient brush
PdfRadialGradientBrush brush = new PdfRadialGradientBrush(new PointF(50,
50), 0, new PointF(50, 50), 50, Color.FromArgb(255, 255, 0, 0),
Color.FromArgb(255, 0, 0, 255));
//Draw ellipse on the page
graphics.DrawEllipse(brush, new RectangleF(0, 0, 100, 100));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the instance of PdfDocument
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "RadialGradientBrush.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF radial gradient brush
PdfRadialGradientBrush brush = new PdfRadialGradientBrush(new PointF(50,
50), 0, new PointF(50, 50), 50, Color.Red, Color.Blue);
//Draw ellipse on the page
graphics.DrawEllipse(brush, new RectangleF(0, 0, 100, 100));
//Save the PDF document stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "RadialGradientBrush.pdf";
return fileStreamResult;
```

XAMARIN

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF radial gradient brush
```

```

PdfRadialGradientBrush brush = new PdfRadialGradientBrush(new PointF(50,
50), 0, new PointF(50, 50), 50, Syncfusion.Drawing.Color.Red,
Syncfusion.Drawing.Color.Blue);
//Draw ellipse on the page
graphics.DrawEllipse(brush, new RectangleF(0, 0, 100, 100));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the instance of PdfDocument
doc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("RadialGradientBrush.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("RadialGradientBrush.pdf", "application/pdf", stream);
}

```

Tiling Brush

The tiling brush is used to draw an object repeatedly. You can draw any shape on PDF page with tiling brush using [PdfTilingBrush](#). The following code snippet explains this.

C#

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF tiling brush
PdfTilingBrush brush = new PdfTilingBrush(new RectangleF(0, 0, 11, 11));
//Draw ellipse inside the tile
brush.Graphics.DrawEllipse(PdfPens.Red, new RectangleF(0, 0, 10, 10));
//Draw ellipse
graphics.DrawEllipse(brush, new RectangleF(0, 0, 200, 100));
//Save the PDF document
doc.Save("TilingBrush.pdf");
//Close the instance of PdfDocument
doc.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim doc As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = doc.Pages.Add

```

```

'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create new PDF tiling brush
Dim brush As PdfTilingBrush = New PdfTilingBrush(New RectangleF(0, 0, 11,
11))
'Draw ellipse inside the tile
brush.Graphics.DrawEllipse(PdfPens.Red, New RectangleF(0, 0, 10, 10))
'Draw ellipse
graphics.DrawEllipse(brush, New RectangleF(0, 0, 200, 100))
'Save the PDF document
doc.Save("TilingBrush.pdf")
'Close the instance of PdfDocument
doc.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF tiling brush
PdfTilingBrush brush = new PdfTilingBrush(new RectangleF(0, 0, 11, 11));
//Draw ellipse inside the tile
brush.Graphics.DrawEllipse(PdfPens.Red, new RectangleF(0, 0, 10, 10));
//Draw ellipse
graphics.DrawEllipse(brush, new RectangleF(0, 0, 200, 100));
//Save the PDF document as stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the instance of PdfDocument
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "TilingBrush.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF tiling brush
PdfTilingBrush brush = new PdfTilingBrush(new RectangleF(0, 0, 11, 11));
//Draw ellipse inside the tile
brush.Graphics.DrawEllipse(PdfPens.Red, new RectangleF(0, 0, 10, 10));
//Draw ellipse
graphics.DrawEllipse(brush, new RectangleF(0, 0, 200, 100));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Set the position as '0'
stream.Position = 0;

```

```
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileName = "TilingBrush.pdf";
return fileStreamResult;
```

XAMARIN

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create new PDF tiling brush
PdfTilingBrush brush = new PdfTilingBrush(new RectangleF(0, 0, 11, 11));
//Draw ellipse inside the tile
brush.Graphics.DrawEllipse(PdfPens.Red, new RectangleF(0, 0, 10, 10));
//Draw ellipse
graphics.DrawEllipse(brush, new RectangleF(0, 0, 200, 100));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the instance of PdfDocument
doc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("TilingBrush.p
df", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("TilingBrush.pdf",
"application/pdf", stream);
}
```

Working with Tables

Essential PDF provides support for two types of table models, both having a different levels of customization, which is explained below. The two types of table models are

1. [PdfGrid](#)
2. [PdfLightTable](#)

Creating a simple table

Creating a simple table using PdfLightTable in a new document

Essential PDF allows you to create the table with [DataSource](#) from DataSet, Data Table, arrays and IEnumerable objects using [PdfLightTable](#) class. It allows you to perform simple formatting.

Note: In Silverlight, Windows store apps and Xamarin only strongly typed IEnumerable objects are supported.

The below code snippet illustrates how to create a simple table from a data source using PdfLightTable.

C#

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
// Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
// Initialize DataTable to assign as DataSource to the light table.
DataTable table = new DataTable();
//Include columns to the DataTable.
table.Columns.Add("Name");
table.Columns.Add("Age");
table.Columns.Add("Sex");
//Include rows to the DataTable.
table.Rows.Add(new string[] { "abc", "21", "Male" });
//Assign data source.
pdfLightTable.DataSource = table;
//Draw PdfLightTable.
pdfLightTable.Draw(page, new PointF(0, 0));
//Save the document.
doc.Save("Output.pdf");
//Close the document
doc.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page.
Dim page As PdfPage = doc.Pages.Add()
' Create a PdfLightTable.
Dim pdfLightTable As New PdfLightTable()
' Initialize DataTable to assign as DataSource to the light table.
Dim table As New DataTable()
'Include columns to the DataTable.
table.Columns.Add("Name")
table.Columns.Add("Age")
table.Columns.Add("Sex")
'Include rows to the DataTable.
table.Rows.Add(New String() { "abc", "21", "Male" })
'Assign data source.
pdfLightTable.DataSource = table
'Draw PdfLightTable.
pdfLightTable.Draw(page, New PointF(0, 0))
'Save the document.
doc.Save("Output.pdf")
'Close the document
doc.Close(True)
```

UWP


```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
// Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
object row = new { Name = "abc", Age = "21", Sex = "Male" };
data.Add(row);
//Add list to IEnumerable
IEnumerable<object> table = data;
//Assign data source.
pdfLightTable.DataSource = table;
//Draw PdfLightTable.
pdfLightTable.Draw(page, new PointF(0, 0));
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
// Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
object row = new { Name = "abc", Age = "21", Sex = "Male" };
data.Add(row);
//Add list to IEnumerable
IEnumerable<object> table = data;
//Assign data source.
pdfLightTable.DataSource = table;
//Draw PdfLightTable.
pdfLightTable.Draw(page, new Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream.
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";

```

```
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
// Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
object row = new { Name = "abc", Age = "21", Sex = "Male" };
data.Add(row);
//Add list to IEnumerable
IEnumerable<object> table = data;
//Assign data source.
pdfLightTable.DataSource = table;
//Draw PdfLightTable.
pdfLightTable.Draw(page, new Syncfusion.Drawing.PointF(0, 0));
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

You can directly add rows and columns instead of a data source, by setting [DataSourceType](#) property to **TableDirect** of [PdfLightTableDataSourceType](#) Enum.

The following code illustrates how to add the data directly into the [PdfLightTable](#).

C#

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Acquire page's graphics.
PdfGraphics graphics = page.Graphics;
```

```
//Declare a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the Data source as direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
//Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty);
//Save the document.
doc.Save("Output.pdf");
//Close the document
doc.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page.
Dim page As PdfPage = doc.Pages.Add()
'Acquire page's graphics.
Dim graphics As PdfGraphics = page.Graphics
'Declare a PdfLightTable.
Dim pdfLightTable As New PdfLightTable()
'Set the Data source type as direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect
'Create columns.
pdfLightTable.Columns.Add(New PdfColumn("Roll Number"))
pdfLightTable.Columns.Add(New PdfColumn("Name"))
pdfLightTable.Columns.Add(New PdfColumn("Class"))
'Add rows.
pdfLightTable.Rows.Add(New Object() {"111", "Maxim", "III"})
'Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty)
'Save the document.
doc.Save("Output.pdf")
'Close the document
doc.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Acquire page's graphics.
PdfGraphics graphics = page.Graphics;
//Declare a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the Data source as direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
```

```
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
//Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Acquire page's graphics.
PdfGraphics graphics = page.Graphics;
//Declare a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the Data source as direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
//Draw the PdfLightTable.
pdfLightTable.Draw(page, Syncfusion.Drawing.PointF.Empty);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream.
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
```

```

PdfPage page = doc.Pages.Add();
//Acquire page's graphics.
PdfGraphics graphics = page.Graphics;
//Declare a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the Data source as direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
//Draw the PdfLightTable.
pdfLightTable.Draw(page, Syncfusion.Drawing.PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Creating a simple table using PdfLightTable in an existing document

You can create table using [PdfLightTable](#) in the existing document by using the following code sample.

C#

```

//Load a PDF document.
PdfLoadedDocument doc = new PdfLoadedDocument("input.pdf");
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
// Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
// Initialize DataTable to assign as DataSource to the light table.
DataTable table = new DataTable();
//Include columns to the DataTable.
table.Columns.Add("Name");
table.Columns.Add("Age");
table.Columns.Add("Sex");
//Include rows to the DataTable.

```

```

table.Rows.Add(new string[] { "abc", "21", "Male" });
//Assign data source.
pdfLightTable.DataSource = table;
//Draw PdfLightTable.
pdfLightTable.Draw(graphics, new PointF(0, 0));
//Save the document.
doc.Save("Output.pdf");
//Close the document
doc.Close(true);

```

VB.NET

```

'Load a PDF document.
Dim doc As New PdfLoadedDocument("input.pdf")
'Get first page from document
Dim page As PdfLoadedPage = TryCast(doc.Pages(0), PdfLoadedPage)
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
' Create a PdfLightTable.
Dim pdfLightTable As New PdfLightTable()
' Initialize DataTable to assign as DataSource to the light table.
Dim table As New DataTable()
'Include columns to the DataTable.
table.Columns.Add("Name")
table.Columns.Add("Age")
table.Columns.Add("Sex")
'Include rows to the DataTable.
table.Rows.Add(New String() {"abc", "21", "Male"})
'Assign data source.
pdfLightTable.DataSource = table
'Draw PdfLightTable.
pdfLightTable.Draw(graphics, New PointF(0, 0))
'Save the document.
doc.Save("Output.pdf")
'Close the document
doc.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument doc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await doc.OpenAsync(file);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list

```

```

List<object> data = new List<object>();
object row = new { Name = "abc", Age = "21", Sex = "Male" };
data.Add(row);
//Add list to IEnumerable
IEnumerable<object> table = data;
//Assign data source.
pdfLightTable.DataSource = table;
//Draw PdfLightTable.
pdfLightTable.Draw(graphics, new PointF(0, 0));
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
// Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
object row = new { Name = "abc", Age = "21", Sex = "Male" };
data.Add(row);
//Add list to IEnumerable
IEnumerable<object> table = data;
//Assign data source.
pdfLightTable.DataSource = table;
//Draw PdfLightTable.
pdfLightTable.Draw(graphics, new Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream.
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
object row = new { Name = "abc", Age = "21", Sex = "Male" };
data.Add(row);
//Add list to IEnumerable
IEnumerable<object> table = data;
//Assign data source.
pdfLightTable.DataSource = table;
//Draw PdfLightTable.
pdfLightTable.Draw(graphics, new Syncfusion.Drawing.PointF(0, 0));
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Creating a simple table using PdfGrid in a new document

[PdfGrid](#) allows you to create table by entering the data manually or from an external data source. The [DataSource](#) can be a data set, data table, arrays or a IEnumerable object.

Note: In Silverlight, Windows store apps and Xamarin only strongly typed IEnumerable objects are supported.

The below code snippet illustrates how to create a simple table from a data source using PdfGrid.

C#

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();

```



```
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable.
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
//Add rows to the DataTable.
dataTable.Rows.Add(new object[] { "E01", "Clay" });
dataTable.Rows.Add(new object[] { "E02", "Thomas" });
//Assign data source.
pdfGrid.DataSource = dataTable;
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new PointF(10, 10));
//Save the document.
doc.Save("Output.pdf");
//close the document
doc.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page.
Dim page As PdfPage = doc.Pages.Add()
' Create a PdfLightTable.
Dim pdfLightTable As New PdfLightTable()
' Initialize DataTable to assign as DataSource to the light table.
Dim table As New DataTable()
'Include columns to the DataTable.
table.Columns.Add("Name")
table.Columns.Add("Age")
table.Columns.Add("Sex")
'Include rows to the DataTable.//you can add multiple rows
table.Rows.Add(New String() {"abc", "21", "Male"})
'Set DataSourceType to the light table.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.External
'Assign data source.
pdfLightTable.DataSource = table
'Draw PdfLightTable.
pdfLightTable.Draw(page, New PointF(0, 0))
'Save the document.
doc.Save("Output.pdf")
'Close the document
doc.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
```

```
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
data.Add(row1);
data.Add(row2);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new PointF(10, 10));
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
data.Add(row1);
data.Add(row2);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new Syncfusion.Drawing.PointF(10, 10));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream.
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
data.Add(row1);
data.Add(row2);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new Syncfusion.Drawing.PointF(10, 10));
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

You can set the data directly without setting any data source using [PdfGridRow](#) and [PdfGridColumn](#) classes.

The below code snippet illustrates how to create the simple table directly using [PdfGrid](#).

C#

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add three columns.
pdfGrid.Columns.Add(3);

```

```
//Add header.
pdfGrid.Headers.Add(1);
PdfGridRow pdfGridHeader = pdfGrid.Headers[0];
pdfGridHeader.Cells[0].Value = "Employee ID";
pdfGridHeader.Cells[1].Value = "Employee Name";
pdfGridHeader.Cells[2].Value = "Salary";
//Add rows.
PdfGridRow pdfGridRow = pdfGrid.Rows.Add();
pdfGridRow.Cells[0].Value = "E01";
pdfGridRow.Cells[1].Value = "Clay";
pdfGridRow.Cells[2].Value = "$10,000";
//Draw the PdfGrid.
pdfGrid.Draw(pdfPage, PointF.Empty);
//Save the document.
pdfDocument.Save("Output.pdf");
//Close the document
pdfDocument.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim pdfDocument As New PdfDocument()
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
'Create a new PdfGrid.
Dim pdfGrid As New PdfGrid()
'Add three columns.
pdfGrid.Columns.Add(3)
'Add header.
pdfGrid.Headers.Add(1)
Dim pdfGridHeader As PdfGridRow = pdfGrid.Headers(0)
pdfGridHeader.Cells(0).Value = "Employee ID"
pdfGridHeader.Cells(1).Value = "Employee Name"
pdfGridHeader.Cells(2).Value = "Salary"
'Add rows.
Dim pdfGridRow As PdfGridRow = pdfGrid.Rows.Add()
pdfGridRow.Cells(0).Value = "E01"
pdfGridRow.Cells(1).Value = "Clay"
pdfGridRow.Cells(2).Value = "$10,000"
'Draw the PdfGrid.
pdfGrid.Draw(pdfPage, PointF.Empty)
'Save the document.
pdfDocument.Save("Output.pdf")
'Close the document
pdfDocument.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add three columns.
pdfGrid.Columns.Add(3);
//Add header.
pdfGrid.Headers.Add(1);
```

```

PdfGridRow pdfGridHeader = pdfGrid.Headers[0];
pdfGridHeader.Cells[0].Value = "Employee ID";
pdfGridHeader.Cells[1].Value = "Employee Name";
pdfGridHeader.Cells[2].Value = "Salary";
//Add rows.
PdfGridRow pdfGridRow = pdfGrid.Rows.Add();
pdfGridRow.Cells[0].Value = "E01";
pdfGridRow.Cells[1].Value = "Clay";
pdfGridRow.Cells[2].Value = "$10,000";
//Draw the PdfGrid.
pdfGrid.Draw(pdfPage, PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
//Close the document.
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add three columns.
pdfGrid.Columns.Add(3);
//Add header.
pdfGrid.Headers.Add(1);
PdfGridRow pdfGridHeader = pdfGrid.Headers[0];
pdfGridHeader.Cells[0].Value = "Employee ID";
pdfGridHeader.Cells[1].Value = "Employee Name";
pdfGridHeader.Cells[2].Value = "Salary";
//Add rows.
PdfGridRow pdfGridRow = pdfGrid.Rows.Add();
pdfGridRow.Cells[0].Value = "E01";
pdfGridRow.Cells[1].Value = "Clay";
pdfGridRow.Cells[2].Value = "$10,000";
//Draw the PdfGrid.
pdfGrid.Draw(pdfPage, Syncfusion.Drawing.PointF.Empty);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream.
pdfDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
pdfDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.

```

```
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add three columns.
pdfGrid.Columns.Add(3);
//Add header.
pdfGrid.Headers.Add(1);
PdfGridRow pdfGridHeader = pdfGrid.Headers[0];
pdfGridHeader.Cells[0].Value = "Employee ID";
pdfGridHeader.Cells[1].Value = "Employee Name";
pdfGridHeader.Cells[2].Value = "Salary";
//Add rows.
PdfGridRow pdfGridRow = pdfGrid.Rows.Add();
pdfGridRow.Cells[0].Value = "E01";
pdfGridRow.Cells[1].Value = "Clay";
pdfGridRow.Cells[2].Value = "$10,000";
//Draw the PdfGrid.
pdfGrid.Draw(pdfPage, Syncfusion.Drawing.PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Close the document.
pdfDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

You can create table using [PdfGrid](#) by loading the IEnumerable data source. Refer to the following code.

C#

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page
PdfPage page = doc.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
```

```

List<object> data = new List<object>();
Object row1 = new { ID = "1", Name = "Clay" };
Object row2 = new { ID = "2", Name = "Gray" };
Object row3 = new { ID = "3", Name = "Ash" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
//Add list to IEnumerable
IEnumerable<object> tableData = data;
//Assign data source
pdfGrid.DataSource = tableData;
//Draw grid to the page of PDF document
pdfGrid.Draw(page, new PointF(10, 10));
//Save the document
doc.Save("Sample.pdf");
//close the document
doc.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim doc As New PdfDocument()
'Add a page
Dim page As PdfPage = doc.Pages.Add()
'Create a PdfGrid
Dim pdfGrid As New PdfGrid()
'Add values to list
Dim data As New List(Of Object)()
Dim row1 As Object = New With {Key .ID = "1", Key .Name = "Clay"}
Dim row2 As Object = New With {Key .ID = "2", Key .Name = "Gray"}
Dim row3 As Object = New With {Key .ID = "3", Key .Name = "Ash"}
data.Add(row1)
data.Add(row2)
data.Add(row3)
'Add list to IEnumerable
Dim tableData As IEnumerable(Of Object) = data
'Assign data source
pdfGrid.DataSource = tableData
'Draw grid to the page of PDF document
pdfGrid.Draw(page, New PointF(10, 10))
'Save the document
doc.Save("Sample.pdf")
'close the document
doc.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page
PdfPage page = doc.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "1", Name = "Clay" };

```

```

Object row2 = new { ID = "2", Name = "Gray" };
Object row3 = new { ID = "3", Name = "Ash" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
//Add list to IEnumerable
IEnumerable<object> tableData = data;
//Assign data source
pdfGrid.DataSource = tableData;
//Draw grid to the page of PDF document
pdfGrid.Draw(page, new PointF(10, 10));
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page
PdfPage page = doc.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "1", Name = "Clay" };
Object row2 = new { ID = "2", Name = "Gray" };
Object row3 = new { ID = "3", Name = "Ash" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
//Add list to IEnumerable
IEnumerable<object> tableData = data;
//Assign data source
pdfGrid.DataSource = tableData;
//Draw grid to the page of PDF document
pdfGrid.Draw(page, new Syncfusion.Drawing.PointF(10, 10));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream.
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.

```



```
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page
PdfPage page = doc.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "1", Name = "Clay" };
Object row2 = new { ID = "2", Name = "Gray" };
Object row3 = new { ID = "3", Name = "Ash" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
//Add list to IEnumerable
IEnumerable<object> tableData = data;
//Assign data source
pdfGrid.DataSource = tableData;
//Draw grid to the page of PDF document
pdfGrid.Draw(page, new Syncfusion.Drawing.PointF(10, 10));
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Creating a simple table using PdfGrid in an existing document

You can create a table using [PdfGrid](#) in the existing document by using the following code sample.

C#

```
//Load a PDF document.
PdfLoadedDocument doc = new PdfLoadedDocument("input.pdf");
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
```

```
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable.
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
//Add rows to the DataTable.
dataTable.Rows.Add(new object[] { "E01", "Clay" });
dataTable.Rows.Add(new object[] { "E02", "Thomas" });
//Assign data source.
pdfGrid.DataSource = dataTable;
//Draw grid to the page of PDF document.
pdfGrid.Draw(graphics, new PointF(10, 10));
//Save the document.
doc.Save("Output.pdf");
//close the document
doc.Close(true);
```

VB.NET

```
'Load a PDF document.
Dim doc As New PdfLoadedDocument("input.pdf")
'Get first page from document
Dim page As PdfLoadedPage = TryCast(doc.Pages(0), PdfLoadedPage)
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create a PdfGrid.
Dim pdfGrid As New PdfGrid()
'Create a DataTable.
Dim dataTable As New DataTable()
'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
'Add rows to the DataTable.
dataTable.Rows.Add(New Object() {"E01", "Clay"})
dataTable.Rows.Add(New Object() {"E02", "Thomas"})
'Assign data source.
pdfGrid.DataSource = dataTable
'Draw grid to the page of PDF document.
pdfGrid.Draw(graphics, New PointF(10, 10))
'Save the document.
doc.Save("Output.pdf")
'close the document
doc.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument doc = new PdfLoadedDocument();
```

```

//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await doc.OpenAsync(file);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "1", Name = "Clay" };
Object row2 = new { ID = "2", Name = "Thomas" };
data.Add(row1);
data.Add(row2);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Draw grid to the page of PDF document.
pdfGrid.Draw(graphics, new PointF(10, 10));
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open, FileAccess.Read);
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "1", Name = "Clay" };
Object row2 = new { ID = "2", Name = "Thomas" };
data.Add(row1);
data.Add(row2);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Draw grid to the page of PDF document.
pdfGrid.Draw(graphics, new Syncfusion.Drawing.PointF(10, 10));
//Creating the stream object
MemoryStream stream = new MemoryStream();

```

```
//Save the PDF document to stream.
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
//Get first page from document
PdfLoadedPage page = doc.Pages[0] as PdfLoadedPage;
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "1", Name = "Clay" };
Object row2 = new { ID = "2", Name = "Thomas" };
data.Add(row1);
data.Add(row2);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Draw grid to the page of PDF document.
pdfGrid.Draw(graphics, new Syncfusion.Drawing.PointF(10, 10));
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Support for cell customization

Cell customization in PdfLightTable

[PdfLightTable](#) allows users to customize cell font, background, border, etc. using [PdfCellStyle](#) class.

The below code snippet illustrates how to customize the cell properties in PdfLightTable.

C#

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add Rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
pdfLightTable.Rows.Add(new object[] { "112", "Minim", "III" });
//Create the font for setting the style.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Declare and define the alternate style.
PdfCellStyle altStyle = new PdfCellStyle(font, PdfBrushes.White,
PdfPens.Green);
altStyle.BackgroundBrush = PdfBrushes.DarkGray;
//Declare and define the header style.
PdfCellStyle headerStyle = new PdfCellStyle(font, PdfBrushes.White,
PdfPens.Brown);
headerStyle.BackgroundBrush = PdfBrushes.Red;
pdfLightTable.Style.AlternateStyle = altStyle;
pdfLightTable.Style.HeaderStyle = headerStyle;
//Show header in the table
pdfLightTable.Style.ShowHeader = true;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty);
//Save the document.
doc.Save("Output.pdf");
//Close the document
doc.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page.
Dim page As PdfPage = doc.Pages.Add()
'Create a PdfLightTable.
```

```

Dim pdfLightTable As New PdfLightTable()
'Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect
'Create columns.
pdfLightTable.Columns.Add(New PdfColumn("Roll Number"))
pdfLightTable.Columns.Add(New PdfColumn("Name"))
pdfLightTable.Columns.Add(New PdfColumn("Class"))
'Add Rows.
pdfLightTable.Rows.Add(New Object() {"111", "Maxim", "III"})
pdfLightTable.Rows.Add(New Object() {"112", "Minim", "III"})
'Create the font for setting the style.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 14)
'Declare and define the alternate style.
Dim altStyle As New PdfCellStyle(font, PdfBrushes.White, PdfPens.Green)
altStyle.BackgroundBrush = PdfBrushes.DarkGray
'Declare and define the header style.
Dim headerStyle As New PdfCellStyle(font, PdfBrushes.White, PdfPens.Brown)
headerStyle.BackgroundBrush = PdfBrushes.Red
pdfLightTable.Style.AlternateStyle = altStyle
pdfLightTable.Style.HeaderStyle = headerStyle
'Show the header in the table
pdfLightTable.Style.ShowHeader = True
'Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty)
'Save the document.
doc.Save("Output.pdf")
'Close the document
doc.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add Rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
pdfLightTable.Rows.Add(new object[] { "112", "Minim", "III" });
//Create the font for setting the style.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Declare and define the alternate style.
PdfCellStyle altStyle = new PdfCellStyle(font, PdfBrushes.White, PdfPens.Green);
altStyle.BackgroundBrush = PdfBrushes.DarkGray;
//Declare and define the header style.
PdfCellStyle headerStyle = new PdfCellStyle(font, PdfBrushes.White, PdfPens.Brown);
headerStyle.BackgroundBrush = PdfBrushes.Red;

```

```
pdfLightTable.Style.AlternateStyle = altStyle;
pdfLightTable.Style.HeaderStyle = headerStyle;
//Show header in the table
pdfLightTable.Style.ShowHeader = true;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add Rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
pdfLightTable.Rows.Add(new object[] { "112", "Minim", "III" });
//Create the font for setting the style.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Declare and define the alternate style.
PdfCellStyle altStyle = new PdfCellStyle(font, PdfBrushes.White,
PdfPens.Green);
altStyle.BackgroundBrush = PdfBrushes.DarkGray;
//Declare and define the header style.
PdfCellStyle headerStyle = new PdfCellStyle(font, PdfBrushes.White,
PdfPens.Brown);
headerStyle.BackgroundBrush = PdfBrushes.Red;
pdfLightTable.Style.AlternateStyle = altStyle;
pdfLightTable.Style.HeaderStyle = headerStyle;
//Show header in the table
pdfLightTable.Style.ShowHeader = true;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, Syncfusion.Drawing.PointF.Empty);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream.
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
```

```
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add Rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
pdfLightTable.Rows.Add(new object[] { "112", "Minim", "III" });
//Create the font for setting the style.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
//Declare and define the alternate style.
PdfCellStyle altStyle = new PdfCellStyle(font, PdfBrushes.White,
PdfPens.Green);
altStyle.BackgroundBrush = PdfBrushes.DarkGray;
//Declare and define the header style.
PdfCellStyle headerStyle = new PdfCellStyle(font, PdfBrushes.White,
PdfPens.Brown);
headerStyle.BackgroundBrush = PdfBrushes.Red;
pdfLightTable.Style.AlternateStyle = altStyle;
pdfLightTable.Style.HeaderStyle = headerStyle;
//Show header in the table
pdfLightTable.Style.ShowHeader = true;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, Syncfusion.Drawing.PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
```



```
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

You can set different styles for particular cell using [BeginCellLayout](#) and [EndCellLayout](#) events in [PdfLightTable](#) class.

The following code example illustrates how to draw the graphics elements in particular cell using these event handlers.

C#

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
pdfLightTable.Rows.Add(new object[] { "112", "Minim", "III" });
//Subscribing to events
pdfLightTable.BeginCellLayout += pdfLightTable_BeginCellLayout;
pdfLightTable.EndCellLayout += pdfLightTable_EndCellLayout;
//Show the header.
pdfLightTable.Style.ShowHeader = true;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty);
//Save the document.
doc.Save("Output.pdf");
//Close the document
doc.Close(true);
private void pdfLightTable_EndCellLayout(object sender,
EndCellLayoutEventArgs args)
{
if(args.RowIndex==0 &&args.CellIndex==0)
{
args.Graphics.DrawImage(new PdfBitmap(imageFileName), args.Bounds);
}
}
private void pdfLightTable_BeginCellLayout(object sender,
BeginCellLayoutEventArgs args)
{
if (args.RowIndex == 0 && args.CellIndex == 1)
{
args.Graphics.DrawEllipse(PdfBrushes.Red, args.Bounds);
}
}
}
```

VB.NET

```

'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page.
Dim page As PdfPage = doc.Pages.Add()
'Declare a PdfLightTable
Dim pdfLightTable As New PdfLightTable()
'Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect
'Create columns.
pdfLightTable.Columns.Add(New PdfColumn("Roll Number"))
pdfLightTable.Columns.Add(New PdfColumn("Name"))
pdfLightTable.Columns.Add(New PdfColumn("Class"))
'Add rows.
pdfLightTable.Rows.Add(New Object() {"111", "Maxim", "III"})
pdfLightTable.Rows.Add(New Object() {"112", "Minim", "III"})
'Subscribing to events
AddHandler pdfLightTable.BeginCellLayout, AddressOf
pdfLightTable_BeginCellLayout
AddHandler pdfLightTable.EndCellLayout, AddressOf
pdfLightTable_EndCellLayout
'Show the header.
pdfLightTable.Style.ShowHeader = True
'Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty)
'Save the document.
doc.Save("Output.pdf")
'Close the document
doc.Close(True)
Private Sub pdfLightTable_EndCellLayout(ByVal sender As Object, ByVal args
As EndCellLayoutEventArgs)
If args.RowIndex = 0 AndAlso args.CellIndex = 0 Then
args.Graphics.DrawImage(New PdfBitmap(imageFileName), args.Bounds)
End If
End Sub
Private Sub pdfLightTable_BeginCellLayout(ByVal sender As Object, ByVal args
As BeginCellLayoutEventArgs)
If args.RowIndex = 0 AndAlso args.CellIndex = 1 Then
args.Graphics.DrawEllipse(PdfBrushes.Red, args.Bounds)
End If
End Sub

```

UWP

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));

```

```

pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
pdfLightTable.Rows.Add(new object[] { "112", "Minim", "III" });
//Subscribing to events
pdfLightTable.BeginCellLayout += pdfLightTable_BeginCellLayout;
pdfLightTable.EndCellLayout += pdfLightTable_EndCellLayout;
//Show the header.
pdfLightTable.Style.ShowHeader = true;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
private void pdfLightTable_EndCellLayout(object sender,
EndCellLayoutEventArgs args)
{
    if (args.RowIndex == 0 && args.CellIndex == 0)
    {
        //Load the PDF document as stream
        Stream imageStream =
        typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Image.jpg");
        args.Graphics.DrawImage(new PdfBitmap(imageStream), args.Bounds);
    }
}
private void pdfLightTable_BeginCellLayout(object sender,
BeginCellLayoutEventArgs args)
{
    if (args.RowIndex == 0 && args.CellIndex == 1)
    {
        args.Graphics.DrawEllipse(PdfBrushes.Red, args.Bounds);
    }
}

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.

```

```

pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
pdfLightTable.Rows.Add(new object[] { "112", "Minim", "III" });
//Subscribing to events
pdfLightTable.BeginCellLayout += pdfLightTable_BeginCellLayout;
pdfLightTable.EndCellLayout += pdfLightTable_EndCellLayout;
//Show the header.
pdfLightTable.Style.ShowHeader = true;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, Syncfusion.Drawing.PointF.Empty);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream.
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
private void pdfLightTable_EndCellLayout(object sender,
EndCellLayoutEventArgs args)
{
    if (args.RowIndex == 0 && args.CellIndex == 0)
    {
        //Load the image as stream
        FileStream imageStream = new FileStream("Image.jpg", FileMode.Open,
        FileAccess.Read);
        args.Graphics.DrawImage(new PdfBitmap(imageStream), args.Bounds);
    }
}
private void pdfLightTable_BeginCellLayout(object sender,
BeginCellLayoutEventArgs args)
{
    if (args.RowIndex == 0 && args.CellIndex == 1)
    {
        args.Graphics.DrawEllipse(PdfBrushes.Red, args.Bounds);
    }
}

```

XAMARIN

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));

```

```

pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
pdfLightTable.Rows.Add(new object[] { "112", "Minim", "III" });
//Subscribing to events
pdfLightTable.BeginCellLayout += pdfLightTable_BeginCellLayout;
pdfLightTable.EndCellLayout += pdfLightTable_EndCellLayout;
//Show the header.
pdfLightTable.Style.ShowHeader = true;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, Syncfusion.Drawing.PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
private void pdfLightTable_EndCellLayout(object sender,
EndCellLayoutEventArgs args)
{
if (args.RowIndex == 0 && args.CellIndex == 0)
{
//Load the image as stream
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Image.jpg");
args.Graphics.DrawImage(new PdfBitmap(imageStream), args.Bounds);
}
}
private void pdfLightTable_BeginCellLayout(object sender,
BeginCellLayoutEventArgs args)
{
if (args.RowIndex == 0 && args.CellIndex == 1)
{
args.Graphics.DrawEllipse(PdfBrushes.Red, args.Bounds);
}
}
}

```

Cell customization in PdfGrid

[PdfGridCell](#) provides various direct options to customize cells like [ColumnSpan](#), [RowSpan](#), text color, background color, and etc.

The following code example illustrates you how to customize the cell in [PdfGrid](#).

C#

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Create the page
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create the parent grid
PdfGrid parentPdfGrid = new PdfGrid();
//Add the rows
PdfGridRow row1 = parentPdfGrid.Rows.Add();
PdfGridRow row2 = parentPdfGrid.Rows.Add();
row2.Height = 58;
//Add the columns
parentPdfGrid.Columns.Add(3);
//Set the value to the specific cell.
parentPdfGrid.Rows[0].Cells[0].Value = "Nested Table";
parentPdfGrid.Rows[0].Cells[1].RowSpan = 2;
parentPdfGrid.Rows[0].Cells[1].ColumnSpan = 2;
//Create the child table
PdfGrid childPdfGrid = new PdfGrid();
//Set the column and rows for child grid
childPdfGrid.Columns.Add(5);
for (int i = 0; i < 5; i++)
{
    PdfGridRow row = childPdfGrid.Rows.Add();
    for (int j = 0; j < 5; j++)
    {
        row.Cells[j].Value = String.Format("Cell [{0} {1}]", j, i);
    }
}
//Set the value as another PdfGrid in a cell.
parentPdfGrid.Rows[0].Cells[1].Value = childPdfGrid;
//Specify the style for the PdfGridCell.
PdfGridCellStyle pdfGridCellStyle = new PdfGridCellStyle();
pdfGridCellStyle.TextPen = PdfPens.Red;
pdfGridCellStyle.Borders.All = PdfPens.Red;
pdfGridCellStyle.BackgroundImage = new PdfBitmap(imagePath);
PdfGridCell pdfGridCell = parentPdfGrid.Rows[0].Cells[0];
//Apply style
pdfGridCell.Style = pdfGridCellStyle;
//Set image position for the background image in the style.
pdfGridCell.ImagePosition = PdfGridImagePosition.Fit;
//Draw the PdfGrid.
parentPdfGrid.Draw(pdfPage, PointF.Empty);
//Save the document.
pdfDocument.Save("Output.pdf");
//close the document
pdfDocument.Close(true);
```

VB.NET

```

'Create a new PDF document.
Dim pdfDocument As New PdfDocument()
'Create the page
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
'Create the parent grid
Dim parentPdfGrid As New PdfGrid()
'Add the rows
Dim row1 As PdfGridRow = parentPdfGrid.Rows.Add()
Dim row2 As PdfGridRow = parentPdfGrid.Rows.Add()
row2.Height = 58
'Add the columns
parentPdfGrid.Columns.Add(3)
'Set the value to the specific cell.
parentPdfGrid.Rows(0).Cells(0).Value = "Nested Table"
parentPdfGrid.Rows(0).Cells(1).RowSpan = 2
parentPdfGrid.Rows(0).Cells(1).ColumnSpan = 2
'Create the child table
Dim childPdfGrid As New PdfGrid()
'Set the column and rows for child grid
childPdfGrid.Columns.Add(5)
For i As Integer = 0 To 4
Dim row As PdfGridRow = childPdfGrid.Rows.Add()
For j As Integer = 0 To 4
row.Cells(j).Value = String.Format("Cell [{0} {1}]", j, i)
Next j
Next i
'Set the value as another PdfGrid in a cell.
parentPdfGrid.Rows(0).Cells(1).Value = childPdfGrid
'Specify the style for the PdfGridCell.
Dim pdfGridCellStyle As New PdfGridCellStyle()
pdfGridCellStyle.TextPen = PdfPens.Red
pdfGridCellStyle.Borders.All = PdfPens.Red
pdfGridCellStyle.BackgroundImage = New PdfBitmap(imageFileName)
Dim pdfGridCell As PdfGridCell = parentPdfGrid.Rows(0).Cells(0)
'Apply style
pdfGridCell.Style = pdfGridCellStyle
'Set image position for the background image in the style.
pdfGridCell.ImagePosition = PdfGridImagePosition.Fit
'Draw the PdfGrid.
parentPdfGrid.Draw(pdfPage, PointF.Empty)
'Save the document.
pdfDocument.Save("Output.pdf")
'close the document
pdfDocument.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Create the page
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create the parent grid
PdfGrid parentPdfGrid = new PdfGrid();
//Add the rows
PdfGridRow row1 = parentPdfGrid.Rows.Add();
PdfGridRow row2 = parentPdfGrid.Rows.Add();

```

```

row2.Height = 58;
//Add the columns
parentPdfGrid.Columns.Add(3);
//Set the value to the specific cell.
parentPdfGrid.Rows[0].Cells[0].Value = "Nested Table";
parentPdfGrid.Rows[0].Cells[1].RowSpan = 2;
parentPdfGrid.Rows[0].Cells[1].ColumnSpan = 2;
//Create the child table
PdfGrid childPdfGrid = new PdfGrid();
//Set the column and rows for child grid
childPdfGrid.Columns.Add(5);
for (int i = 0; i < 5; i++)
{
    PdfGridRow row = childPdfGrid.Rows.Add();
    for (int j = 0; j < 5; j++)
    {
        row.Cells[j].Value = String.Format("Cell [{0} {1}]", j, i);
    }
}
//Set the value as another PdfGrid in a cell.
parentPdfGrid.Rows[0].Cells[1].Value = childPdfGrid;
//Specify the style for the PdfGridCell.
PdfGridCellStyle pdfGridCellStyle = new PdfGridCellStyle();
pdfGridCellStyle.TextPen = PdfPens.Red;
pdfGridCellStyle.Borders.All = PdfPens.Red;
//Load the image as stream
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Image.jpg");
pdfGridCellStyle.BackgroundImage = new PdfBitmap(imageStream);
PdfGridCell pdfGridCell = parentPdfGrid.Rows[0].Cells[0];
//Apply style
pdfGridCell.Style = pdfGridCellStyle;
//Set image position for the background image in the style.
pdfGridCell.ImagePosition = PdfGridImagePosition.Fit;
//Draw the PdfGrid.
parentPdfGrid.Draw(pdfPage, PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
//Close the document.
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Create the page
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create the parent grid
PdfGrid parentPdfGrid = new PdfGrid();
//Add the rows
PdfGridRow row1 = parentPdfGrid.Rows.Add();

```



```

PdfGridRow row2 = parentPdfGrid.Rows.Add();
row2.Height = 58;
//Add the columns
parentPdfGrid.Columns.Add(3);
//Set the value to the specific cell.
parentPdfGrid.Rows[0].Cells[0].Value = "Nested Table";
parentPdfGrid.Rows[0].Cells[1].RowSpan = 2;
parentPdfGrid.Rows[0].Cells[1].ColumnSpan = 2;
//Create the child table
PdfGrid childPdfGrid = new PdfGrid();
//Set the column and rows for child grid
childPdfGrid.Columns.Add(5);
for (int i = 0; i < 5; i++)
{
    PdfGridRow row = childPdfGrid.Rows.Add();
    for (int j = 0; j < 5; j++)
    {
        row.Cells[j].Value = String.Format("Cell [{0} {1}]", j, i);
    }
}
//Set the value as another PdfGrid in a cell.
parentPdfGrid.Rows[0].Cells[1].Value = childPdfGrid;
//Specify the style for the PdfGridCell.
PdfGridCellStyle pdfGridCellStyle = new PdfGridCellStyle();
pdfGridCellStyle.TextPen = PdfPens.Red;
pdfGridCellStyle.Borders.All = PdfPens.Red;
//Load image as stream
FileStream imageStream = new FileStream("Image.jpg", FileMode.Open,
    FileAccess.Read);
pdfGridCellStyle.BackgroundImage = new PdfBitmap(imageStream);
PdfGridCell pdfGridCell = parentPdfGrid.Rows[0].Cells[0];
//Apply style
pdfGridCell.Style = pdfGridCellStyle;
//Set image position for the background image in the style.
pdfGridCell.ImagePosition = PdfGridImagePosition.Fit;
//Draw the PdfGrid.
parentPdfGrid.Draw(pdfPage, Syncfusion.Drawing.PointF.Empty);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream.
pdfDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
pdfDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```
//Create a new PDF document.
```

```

PdfDocument pdfDocument = new PdfDocument();
//Create the page
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create the parent grid
PdfGrid parentPdfGrid = new PdfGrid();
//Add the rows
PdfGridRow row1 = parentPdfGrid.Rows.Add();
PdfGridRow row2 = parentPdfGrid.Rows.Add();
row2.Height = 58;
//Add the columns
parentPdfGrid.Columns.Add(3);
//Set the value to the specific cell.
parentPdfGrid.Rows[0].Cells[0].Value = "Nested Table";
parentPdfGrid.Rows[0].Cells[1].RowSpan = 2;
parentPdfGrid.Rows[0].Cells[1].ColumnSpan = 2;
//Create the child table
PdfGrid childPdfGrid = new PdfGrid();
//Set the column and rows for child grid
childPdfGrid.Columns.Add(5);
for (int i = 0; i < 5; i++)
{
    PdfGridRow row = childPdfGrid.Rows.Add();
    for (int j = 0; j < 5; j++)
    {
        row.Cells[j].Value = String.Format("Cell [{0} {1}]", j, i);
    }
}
//Set the value as another PdfGrid in a cell.
parentPdfGrid.Rows[0].Cells[1].Value = childPdfGrid;
//Specify the style for the PdfGridCell.
PdfGridCellStyle pdfGridCellStyle = new PdfGridCellStyle();
pdfGridCellStyle.TextPen = PdfPens.Red;
pdfGridCellStyle.Borders.All = PdfPens.Red;
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Image.jpg");
pdfGridCellStyle.BackgroundImage = new PdfBitmap(imageStream);
PdfGridCell pdfGridCell = parentPdfGrid.Rows[0].Cells[0];
//Apply style
pdfGridCell.Style = pdfGridCellStyle;
//Set image position for the background image in the style.
pdfGridCell.ImagePosition = PdfGridImagePosition.Fit;
//Draw the PdfGrid.
parentPdfGrid.Draw(pdfPage, Syncfusion.Drawing.PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Close the document.
pdfDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Support for rows and columns customization

Row customization in PdfLightTable

[PdfLightTable](#) doesn't provide direct support for row customizations. However, this can be done through the event handlers. The following code snippet illustrates how to customize the row in PdfLightTable using [BeginRowLayout](#) and [EndRowLayout](#) event handlers.

C#

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
pdfLightTable.Rows.Add(new object[] { "112", "Minim", "III" });
pdfLightTable.Rows.Add(new object[] { "113", "john", "III" });
pdfLightTable.Rows.Add(new object[] { "114", "peter", "III" });
//Subscribing to events
pdfLightTable.BeginRowLayout+=pdfLightTable_BeginRowLayout;
pdfLightTable.EndRowLayout+=pdfLightTable_EndRowLayout;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty);
//Save the document.
doc.Save("Output.pdf");
//Close the document
doc.Close(true);
private void pdfLightTable_EndRowLayout(object sender, EndRowLayoutEventArgs
args)
{
//Customize the rows when row layout ends
if (args.RowIndex == 3)
args.Cancel = true;
}
private void pdfLightTable_BeginRowLayout(object sender,
BeginRowLayoutEventArgs args)
{
//Apply column span
if (args.RowIndex==1)
```

```

{
    PdfLightTable table = (PdfLightTable)sender;
    int count = table.Columns.Count;
    int[] spanMap = new int[count];
    // Set just spanned cells. Negative values are not allowed.
    spanMap[0] = 2;
    spanMap[1] = 3;
    args.ColumnSpanMap = spanMap;
}
}

```

VB.NET

```

'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page.
Dim page As PdfPage = doc.Pages.Add()
'Create a PdfLightTable
Dim pdfLightTable As New PdfLightTable()
'Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect
'Create columns.
pdfLightTable.Columns.Add(New PdfColumn("Roll Number"))
pdfLightTable.Columns.Add(New PdfColumn("Name"))
pdfLightTable.Columns.Add(New PdfColumn("Class"))
'Add rows.
pdfLightTable.Rows.Add(New Object() { "111", "Maxim", "III" })
pdfLightTable.Rows.Add(New Object() { "112", "Minim", "III" })
pdfLightTable.Rows.Add(New Object() { "113", "john", "III" })
pdfLightTable.Rows.Add(New Object() { "114", "peter", "III" })
'Subscribing to events
AddHandler pdfLightTable.BeginRowLayout, AddressOf
pdfLightTable_BeginRowLayout
AddHandler pdfLightTable.EndRowLayout, AddressOf pdfLightTable_EndRowLayout
'Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty)
'Save the document.
doc.Save("Output.pdf")
'Close the document
doc.Close(True)
Private Sub pdfLightTable_EndRowLayout(ByVal sender As Object, ByVal args As
EndRowLayoutEventArgs)
'Customize the rows when row layout ends
If args.RowIndex = 3 Then
args.Cancel = True
End If
End Sub
Private Sub pdfLightTable_BeginRowLayout(ByVal sender As Object, ByVal args
As BeginRowLayoutEventArgs)
'Apply column span
If args.RowIndex=1 Then
Dim table As PdfLightTable = CType(sender, PdfLightTable)
Dim count As Integer = table.Columns.Count
Dim spanMap(count - 1) As Integer
' Set just spanned cells. Negative values are not allowed.
spanMap(0) = 2

```

```
spanMap(1) = 3
args.ColumnSpanMap = spanMap
End If
End Sub
```

UWP

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
pdfLightTable.Rows.Add(new object[] { "112", "Minim", "III" });
pdfLightTable.Rows.Add(new object[] { "113", "john", "III" });
pdfLightTable.Rows.Add(new object[] { "114", "peter", "III" });
//Subscribing to events
pdfLightTable.BeginRowLayout += pdfLightTable_BeginRowLayout;
pdfLightTable.EndRowLayout += pdfLightTable_EndRowLayout;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
private void pdfLightTable_EndRowLayout(object sender, EndRowLayoutEventArgs
args)
{
    //Customize the rows when row layout ends
    if (args.RowIndex == 3)
    args.Cancel = true;
}
private void pdfLightTable_BeginRowLayout(object sender,
BeginRowLayoutEventArgs args)
{
    //Apply column span
    if (args.RowIndex == 1)
    {
        PdfLightTable table = (PdfLightTable)sender;
        int count = table.Columns.Count;
        int[] spanMap = new int[count];
        // Set just spanned cells. Negative values are not allowed.
        spanMap[0] = 2;
        spanMap[1] = 3;
```

```
args.ColumnSpanMap = spanMap;
}
}
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
pdfLightTable.Rows.Add(new object[] { "112", "Minim", "III" });
pdfLightTable.Rows.Add(new object[] { "113", "john", "III" });
pdfLightTable.Rows.Add(new object[] { "114", "peter", "III" });
//Subscribing to events
pdfLightTable.BeginRowLayout += pdfLightTable_BeginRowLayout;
pdfLightTable.EndRowLayout += pdfLightTable_EndRowLayout;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, Syncfusion.Drawing.PointF.Empty);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream.
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
private void pdfLightTable_EndRowLayout(object sender, EndRowLayoutEventArgs
args)
{
//Customize the rows when row layout ends
if (args.RowIndex == 3)
args.Cancel = true;
}
private void pdfLightTable_BeginRowLayout(object sender,
BeginRowLayoutEventArgs args)
{
//Apply column span
if (args.RowIndex == 1)
{
```

```

PdfLightTable table = (PdfLightTable)sender;
int count = table.Columns.Count;
int[] spanMap = new int[count];
// Set just spanned cells. Negative values are not allowed.
spanMap[0] = 2;
spanMap[1] = 3;
args.ColumnSpanMap = spanMap;
}
}

```

XAMARIN

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "Maxim", "III" });
pdfLightTable.Rows.Add(new object[] { "112", "Minim", "III" });
pdfLightTable.Rows.Add(new object[] { "113", "john", "III" });
pdfLightTable.Rows.Add(new object[] { "114", "peter", "III" });
//Subscribing to events
pdfLightTable.BeginRowLayout += pdfLightTable_BeginRowLayout;
pdfLightTable.EndRowLayout += pdfLightTable_EndRowLayout;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, Syncfusion.Drawing.PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
}

```

Column customization in PdfLightTable

The following code snippet illustrates how to customize the column in [PdfLightTable](#) using the [PdfStringFormat](#) class.

C#

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Acquire page's graphics.
PdfGraphics graphics = page.Graphics;
//Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "john", "III" });
// Specify column name.
pdfLightTable.Columns[1].ColumnName = "Student Name";
//create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Bottom;
//Apply string format
pdfLightTable.Columns[0].StringFormat = format;
//Style to display header.
pdfLightTable.Style.ShowHeader = true;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty);
//Save the document.
doc.Save("Output.pdf");
//close the document
doc.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page.
Dim page As PdfPage = doc.Pages.Add()
'Acquire page's graphics.
Dim graphics As PdfGraphics = page.Graphics
'Create a PdfLightTable.
Dim pdfLightTable As New PdfLightTable()
'Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect
'Create columns.
pdfLightTable.Columns.Add(New PdfColumn("Roll Number"))
pdfLightTable.Columns.Add(New PdfColumn("Name"))
pdfLightTable.Columns.Add(New PdfColumn("Class"))
'Add rows.
```



```
pdfLightTable.Rows.Add(New Object() { "111", "john", "III" })
' Specify column name.
pdfLightTable.Columns(1).ColumnName = "Student Name"
'create and customize the string format
Dim format As New PdfStringFormat()
format.Alignment = PdfTextAlignment.Center
format.LineAlignment = PdfVerticalAlignment.Bottom
'Apply string format
pdfLightTable.Columns(0).StringFormat = format
'Style to display header.
pdfLightTable.Style.ShowHeader = True
'Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty)
'Save the document.
doc.Save("Output.pdf")
'close the document
doc.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Acquire page's graphics.
PdfGraphics graphics = page.Graphics;
//Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "john", "III" });
// Specify column name.
pdfLightTable.Columns[1].ColumnName = "Student Name";
//create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Bottom;
//Apply string format
pdfLightTable.Columns[0].StringFormat = format;
//Style to display header.
pdfLightTable.Style.ShowHeader = true;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Acquire page's graphics.
PdfGraphics graphics = page.Graphics;
//Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "john", "III" });
// Specify column name.
pdfLightTable.Columns[1].ColumnName = "Student Name";
//create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Bottom;
//Apply string format
pdfLightTable.Columns[0].StringFormat = format;
//Style to display header.
pdfLightTable.Style.ShowHeader = true;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, Syncfusion.Drawing.PointF.Empty);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream.
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Acquire page's graphics.
PdfGraphics graphics = page.Graphics;
//Create a PdfLightTable.
```

```

PdfLightTable pdfLightTable = new PdfLightTable();
//Set the DataSourceType as Direct.
pdfLightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns.
pdfLightTable.Columns.Add(new PdfColumn("Roll Number"));
pdfLightTable.Columns.Add(new PdfColumn("Name"));
pdfLightTable.Columns.Add(new PdfColumn("Class"));
//Add rows.
pdfLightTable.Rows.Add(new object[] { "111", "john", "III" });
// Specify column name.
pdfLightTable.Columns[1].ColumnName = "Student Name";
//create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Bottom;
//Apply string format
pdfLightTable.Columns[0].StringFormat = format;
//Style to display header.
pdfLightTable.Style.ShowHeader = true;
//Draw the PdfLightTable.
pdfLightTable.Draw(page, Syncfusion.Drawing.PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Row customization in PdfGrid

You can customize row height and styles using [Rows](#) property in [PdfGrid](#) class.

The following code snippet illustrates how to customize the row in PdfGrid.

C#

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable.
DataTable dataTable = new DataTable();

```

```

//Add columns to the DataTable.
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
//Add rows to the DataTable.
dataTable.Rows.Add(new object[] { "E01", "John" });
dataTable.Rows.Add(new object[] { "E02", "Thomas" });
dataTable.Rows.Add(new object[] { "E03", "Peter" });
//Assign data source.
pdfGrid.DataSource = dataTable;
//Create an instance of PdfGridRowStyle
PdfGridRowStyle pdfGridRowStyle = new PdfGridRowStyle();
pdfGridRowStyle.BackgroundBrush = PdfBrushes.LightYellow;
pdfGridRowStyle.Font = new PdfStandardFont(PdfFontFamily.Courier, 10);
pdfGridRowStyle.TextBrush = PdfBrushes.Blue;
pdfGridRowStyle.TextPen = PdfPens.Pink;
//Set the height
pdfGrid.Rows[2].Height = 50;
//Set style for the PdfGridRow.
pdfGrid.Rows[0].Style = pdfGridRowStyle;
//Draw the PdfGrid.
PdfGridLayoutResult result = pdfGrid.Draw(pdfPage, PointF.Empty);
//Save the document.
pdfDocument.Save("Output.pdf");
//close the document
pdfDocument.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim pdfDocument As New PdfDocument()
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
'Create a new PdfGrid.
Dim pdfGrid As New PdfGrid()
'Create a DataTable.
Dim dataTable As New DataTable()
'Add columns to the DataTable.
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
'Add rows to the DataTable.
dataTable.Rows.Add(New Object() {"E01", "John"})
dataTable.Rows.Add(New Object() {"E02", "Thomas"})
dataTable.Rows.Add(New Object() {"E03", "Peter"})
'Assign data source.
pdfGrid.DataSource = dataTable
'Create an instance of PdfGridRowStyle
Dim pdfGridRowStyle As New PdfGridRowStyle()
pdfGridRowStyle.BackgroundBrush = PdfBrushes.LightYellow
pdfGridRowStyle.Font = New PdfStandardFont(PdfFontFamily.Courier, 10)
pdfGridRowStyle.TextBrush = PdfBrushes.Blue
pdfGridRowStyle.TextPen = PdfPens.Pink
'Set the height
pdfGrid.Rows(2).Height = 50
'Set style for the PdfGridRow.
pdfGrid.Rows(0).Style = pdfGridRowStyle
'Draw the PdfGrid.
Dim result As PdfGridLayoutResult = pdfGrid.Draw(pdfPage, PointF.Empty)

```

```
'Save the document.
pdfDocument.Save("Output.pdf")
'close the document
pdfDocument.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "John" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E03", Name = "Peter" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Create an instance of PdfGridRowStyle
PdfGridRowStyle pdfGridRowStyle = new PdfGridRowStyle();
pdfGridRowStyle.BackgroundBrush = PdfBrushes.LightYellow;
pdfGridRowStyle.Font = new PdfStandardFont(PdfFontFamily.Courier, 10);
pdfGridRowStyle.TextBrush = PdfBrushes.Blue;
pdfGridRowStyle.TextPen = PdfPens.Pink;
//Set the height
pdfGrid.Rows[2].Height = 50;
//Set style for the PdfGridRow.
pdfGrid.Rows[0].Style = pdfGridRowStyle;
//Draw the PdfGrid.
PdfGridLayoutResult result = pdfGrid.Draw(pdfPage, PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
//Close the document.
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "John" };
Object row2 = new { ID = "E02", Name = "Thomas" };
```

```

Object row3 = new { ID = "E03", Name = "Peter" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Create an instance of PdfGridRowStyle
PdfGridRowStyle pdfGridRowStyle = new PdfGridRowStyle();
pdfGridRowStyle.BackgroundBrush = PdfBrushes.LightYellow;
pdfGridRowStyle.Font = new PdfStandardFont(PdfFontFamily.Courier, 10);
pdfGridRowStyle.TextBrush = PdfBrushes.Blue;
pdfGridRowStyle.TextPen = PdfPens.Pink;
//Set the height
pdfGrid.Rows[2].Height = 50;
//Set style for the PdfGridRow.
pdfGrid.Rows[0].Style = pdfGridRowStyle;
//Draw the PdfGrid.
PdfGridLayoutResult result = pdfGrid.Draw(pdfPage,
Syncfusion.Drawing.PointF.Empty);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream.
pdfDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
pdfDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "John" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E03", Name = "Peter" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;

```

```

//Create an instance of PdfGridRowStyle
PdfGridRowStyle pdfGridRowStyle = new PdfGridRowStyle();
pdfGridRowStyle.BackgroundBrush = PdfBrushes.LightYellow;
pdfGridRowStyle.Font = new PdfStandardFont(PdfFontFamily.Courier, 10);
pdfGridRowStyle.TextBrush = PdfBrushes.Blue;
pdfGridRowStyle.TextPen = PdfPens.Pink;
//Set the height
pdfGrid.Rows[2].Height = 50;
//Set style for the PdfGridRow.
pdfGrid.Rows[0].Style = pdfGridRowStyle;
//Draw the PdfGrid.
PdfGridLayoutResult result = pdfGrid.Draw(pdfPage,
Syncfusion.Drawing.PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Close the document.
pdfDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Columns customization in PdfGrid

You can customize column width and text formats using [Columns](#) property in [PdfGrid](#) class.

The following code snippet illustrates how to customize the column in PdfGrid.

C#

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable.
DataTable dataTable = new DataTable();
//Add columns to the DataTable.
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
//Add rows to the DataTable.
dataTable.Rows.Add(new object[] { "E01", "John" });
dataTable.Rows.Add(new object[] { "E02", "Thomas" });
dataTable.Rows.Add(new object[] { "E03", "Peter" });
//Assign data source.

```

```
pdfGrid.DataSource = dataTable;
//Set the width
pdfGrid.Columns[1].Width = 50;
//create and customize the string formats
PdfStringFormat format=new PdfStringFormat();
format.Alignment=PdfTextAlignment.Center;
format.LineAlignment=PdfVerticalAlignment.Bottom;
//set the column text format
pdfGrid.Columns[0].Format = format;
//Draw the PdfGrid.
PdfGridLayoutResult result = pdfGrid.Draw(pdfPage, PointF.Empty);
//Save the document.
pdfDocument.Save("Output.pdf");
//close the document
pdfDocument.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim pdfDocument As New PdfDocument()
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
'Create a new PdfGrid.
Dim pdfGrid As New PdfGrid()
'Create a DataTable.
Dim dataTable As New DataTable()
'Add columns to the DataTable.
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
'Add rows to the DataTable.
dataTable.Rows.Add(New Object() {"E01", "John"})
dataTable.Rows.Add(New Object() {"E02", "Thomas"})
dataTable.Rows.Add(New Object() {"E03", "Peter"})
'Assign data source.
pdfGrid.DataSource = dataTable
'Set the width
pdfGrid.Columns(1).Width = 50
'create and customize the string formats
Dim format As New PdfStringFormat()
format.Alignment = PdfTextAlignment.Center
format.LineAlignment = PdfVerticalAlignment.Bottom
'set the column text format
pdfGrid.Columns(0).Format = format
'Draw the PdfGrid.
Dim result As PdfGridLayoutResult = pdfGrid.Draw(pdfPage, PointF.Empty)
'Save the document.
pdfDocument.Save("Output.pdf")
'close the document
pdfDocument.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
```



```
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E02", Name = "Peter" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Set the width
pdfGrid.Columns[1].Width = 50;
//create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Bottom;
//set the column text format
pdfGrid.Columns[0].Format = format;
//Draw the PdfGrid.
PdfGridLayoutResult result = pdfGrid.Draw(pdfPage, PointF.Empty);
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
//Close the document.
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E02", Name = "Peter" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Set the width
pdfGrid.Columns[1].Width = 50;
//create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Bottom;
```

```
//set the column text format
pdfGrid.Columns[0].Format = format;
//Draw the PdfGrid.
PdfGridLayoutResult result = pdfGrid.Draw(pdfPage,
Syncfusion.Drawing.PointF.Empty);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream.
pdfDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
pdfDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E02", Name = "Peter" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Set the width
pdfGrid.Columns[1].Width = 50;
//create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Bottom;
//set the column text format
pdfGrid.Columns[0].Format = format;
//Draw the PdfGrid.
PdfGridLayoutResult result = pdfGrid.Draw(pdfPage,
Syncfusion.Drawing.PointF.Empty);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Close the document.
pdfDocument.Close(true);
```

```
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Support for table customization

Table customization in PdfLightTable

Essential PDF supports users to create a customizable PDF table like [CellSpacing](#), [CellPadding](#), [RepeatHeader](#), [ShowHeader](#), etc. This can be achieved by using the [PdfLightTableStyle](#) class.

The following code snippet illustrates how to customize the table using [PdfLightTable](#).

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create a PdfLightTable
PdfLightTable pdfLightTable = new PdfLightTable();
//Initialize DataTable to assign as DataSource to the light table
DataTable table = new DataTable();
//Include columns to the DataTable
table.Columns.Add("Name");
table.Columns.Add("Age");
table.Columns.Add("Sex");
//Include rows to the DataTable
table.Rows.Add(new string[] { "abc", "21", "Male" });
//Assign data source
pdfLightTable.DataSource = table;
//Declare and define light table style
PdfLightTableStyle lightTableStyle = new PdfLightTableStyle();
//Set cell padding, which specifies the space between border and content of
the cell
lightTableStyle.CellPadding = 2;
//Set cell spacing, which specifies the space between the adjacent cells
lightTableStyle.CellSpacing = 2;
//Sets to show header in table
lightTableStyle.ShowHeader = true;
//Sets to repeat header on each page
lightTableStyle.RepeatHeader = true;
//Apply style
pdfLightTable.Style = lightTableStyle;
//Draw PdfLightTable
pdfLightTable.Draw(page, new PointF(0, 0));
```

```
//Save the document
document.Save("Output.pdf");
//Close the document
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As New PdfDocument()
'Add a page
Dim page As PdfPage = document.Pages.Add()
'Create a PdfLightTable
Dim pdfLightTable As New PdfLightTable()
'Initialize DataTable to assign as DataSource to the light table
Dim table As New DataTable()
'Include columns to the DataTable
table.Columns.Add("Name")
table.Columns.Add("Age")
table.Columns.Add("Sex")
'Include rows to the DataTable
table.Rows.Add(New String() {"abc", "21", "Male"})
'Assign data source
pdfLightTable.DataSource = table
'Declare and define light table style
Dim lightTableStyle As New PdfLightTableStyle()
'Set cell padding, which specifies the space between border and content of
the cell
lightTableStyle.CellPadding = 2
'Set cell spacing, which specifies the space between the adjacent cells
lightTableStyle.CellSpacing = 2
'Sets to show header in table
lightTableStyle.ShowHeader = True
'Sets to repeat header on each page
lightTableStyle.RepeatHeader = True
'Apply style
pdfLightTable.Style = lightTableStyle
'Draw PdfLightTable
pdfLightTable.Draw(page, New PointF(0, 0))
'Save the document
document.Save("Output.pdf")
'Close the document
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
// Create a PdfLightTable
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
object row = new { Name = "abc", Age = "21", Sex = "Male" };
data.Add(row);
//Add list to IEnumerable
```

```

IEnumerable<object> table = data;
//Assign data source
pdfLightTable.DataSource = table;
//Declare and define light table style
PdfLightTableStyle lightTableStyle = new PdfLightTableStyle();
//Set cell padding, which specifies the space between border and content of
the cell
lightTableStyle.CellPadding = 2;
//Set cell spacing, which specifies the space between the adjacent cells
lightTableStyle.CellSpacing = 2;
//Sets to show header in table
lightTableStyle.ShowHeader = true;
//Sets to repeat header on each page
lightTableStyle.RepeatHeader = true;
//Apply style
pdfLightTable.Style = lightTableStyle;
//Draw PdfLightTable
pdfLightTable.Draw(page, new PointF(0, 0));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
// Create a PdfLightTable
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
object row = new { Name = "abc", Age = "21", Sex = "Male" };
data.Add(row);
//Add list to IEnumerable
IEnumerable<object> table = data;
//Assign data source
pdfLightTable.DataSource = table;
//Declare and define light table style
PdfLightTableStyle lightTableStyle = new PdfLightTableStyle();
//Set cell padding, which specifies the space between border and content of
the cell
lightTableStyle.CellPadding = 2;
//Set cell spacing, which specifies the space between the adjacent cells
lightTableStyle.CellSpacing = 2;
//Sets to show header in table
lightTableStyle.ShowHeader = true;
//Sets to repeat header on each page
lightTableStyle.RepeatHeader = true;
//Apply style
pdfLightTable.Style = lightTableStyle;

```

```
//Draw PdfLightTable
pdfLightTable.Draw(page, new PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file.
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
// Create a PdfLightTable
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
object row = new { Name = "abc", Age = "21", Sex = "Male" };
data.Add(row);
//Add list to IEnumerable
IEnumerable<object> table = data;
//Assign data source
pdfLightTable.DataSource = table;
//Declare and define light table style
PdfLightTableStyle lightTableStyle = new PdfLightTableStyle();
//Set cell padding, which specifies the space between border and content of
the cell
lightTableStyle.CellPadding = 2;
//Set cell spacing, which specifies the space between the adjacent cells
lightTableStyle.CellSpacing = 2;
//Sets to show header in table
lightTableStyle.ShowHeader = true;
//Sets to repeat header on each page
lightTableStyle.RepeatHeader = true;
//Apply style
pdfLightTable.Style = lightTableStyle;
//Draw PdfLightTable
pdfLightTable.Draw(page, new PointF(0, 0));
//Save the document as stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document instances
document.Close(true);
//Save the stream into PDF file
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.RuntimePlatform == Device.UWP)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
    "application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
    "application/pdf", stream);
}
```

Table customization in PdfGrid

Essential PDF supports users to create a customizable PDF table like [CellSpacing](#), [CellPadding](#), [HorizontalOverflow](#), etc. This can be achieved by using [PdfGridStyle](#) class.

The following code snippet illustrates how to customize the table using [PdfGrid](#).

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "E01", "Clay" });
dataTable.Rows.Add(new object[] { "E02", "Thomas" });
//Assign data source
pdfGrid.DataSource = dataTable;
//Declare and define the grid style
PdfGridStyle gridStyle = new PdfGridStyle();
//Set cell padding, which specifies the space between border and content of
//the cell
gridStyle.CellPadding = new PdfPaddings(2, 2, 2, 2);
//Set cell spacing, which specifies the space between the adjacent cells
gridStyle.CellSpacing = 2;
//Enable to adjust PDF table row width based on the text length
gridStyle.AllowHorizontalOverflow = true;
//Apply style
pdfGrid.Style = gridStyle;
//Draw grid to the page of PDF document
pdfGrid.Draw(page, new PointF(10, 10));
//Save the document
document.Save("Output.pdf");
//Close the document
document.Close(true);
```

VB.NET

```

'Create a new PDF document
Dim document As New PdfDocument()
'Add a page
Dim page As PdfPage = document.Pages.Add()
'Create a PdfGrid
Dim pdfGrid As New PdfGrid()
'Create a DataTable
Dim dataTable As New DataTable()
'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
'Add rows to the DataTable
dataTable.Rows.Add(New Object() {"E01", "Clay"})
dataTable.Rows.Add(New Object() {"E02", "Thomas"})
'Assign data source
pdfGrid.DataSource = dataTable
'Declare and define the grid style
Dim gridStyle As New PdfGridStyle()
'Set cell padding, which specifies the space between border and content of
the cell
gridStyle.CellPadding = New PdfPaddings(2, 2, 2, 2)
'Set cell spacing, which specifies the space between the adjacent cells
gridStyle.CellSpacing = 2
'Enable to adjust PDF table row width based on the text length
gridStyle.AllowHorizontalOverflow = True
'Apply style
pdfGrid.Style = gridStyle
'Draw grid to the page of PDF document
pdfGrid.Draw(page, New PointF(10, 10))
'Save the document
document.Save("Output.pdf")
'Close the document
document.Close(True)
'This will open the PDF file so, the result will be seen in default PDF
viewer
Process.Start("Output.pdf")

```

UWP

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page
PdfPage page = doc.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
data.Add(row1);
data.Add(row2);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source

```



```
pdfGrid.DataSource = dataTable;
//Declare and define the grid style
PdfGridStyle gridStyle = new PdfGridStyle();
//Set cell padding, which specifies the space between border and content of
the cell
gridStyle.CellPadding = new PdfPaddings(2, 2, 2, 2);
//Set cell spacing, which specifies the space between the adjacent cells
gridStyle.CellSpacing = 2;
//Enable to adjust PDF table row width based on the text length
gridStyle.AllowHorizontalOverflow = true;
//Apply style
pdfGrid.Style = gridStyle;
//Draw grid to the page of PDF document
pdfGrid.Draw(page, new PointF(10, 10));
//Save the PDF document into stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
data.Add(row1);
data.Add(row2);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
pdfGrid.DataSource = dataTable;
//Declare and define the grid style
PdfGridStyle gridStyle = new PdfGridStyle();
//Set cell padding, which specifies the space between border and content of
the cell
gridStyle.CellPadding = new PdfPaddings(2, 2, 2, 2);
//Set cell spacing, which specifies the space between the adjacent cells
gridStyle.CellSpacing = 2;
//Enable to adjust PDF table row width based on the text length
gridStyle.AllowHorizontalOverflow = true;
//Apply style
pdfGrid.Style = gridStyle;
//Draw grid to the page of PDF document
pdfGrid.Draw(page, new PointF(10, 10));
//Creating the stream object
MemoryStream stream = new MemoryStream();
```

```
//Save the document as stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file.
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
data.Add(row1);
data.Add(row2);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
pdfGrid.DataSource = dataTable;
//Declare and define the grid style
PdfGridStyle gridStyle = new PdfGridStyle();
//Set cell padding, which specifies the space between border and content of
the cell
gridStyle.CellPadding = new PdfPaddings(2, 2, 2, 2);
//Set cell spacing, which specifies the space between the adjacent cells
gridStyle.CellSpacing = 2;
//Enable to adjust PDF table row width based on the text length
gridStyle.AllowHorizontalOverflow = true;
//Apply style
pdfGrid.Style = gridStyle;
//Draw grid to the page of PDF document
pdfGrid.Draw(page, new PointF(10, 10));
//Save the document as stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document instances
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Built-in table styles

In-built table styles can be applied to both [PdfGrid](#) and [PdfLightTable](#) models and the appearance is made similar to Microsoft Word's built-in table styles. You can also apply in-built table styles with the following additional table style options.

- Banded columns
- Banded rows
- First column
- Last column
- Header row
- Last row

The below code example illustrates how to apply built-in table style using [ApplyBuiltinStyle](#) method of the PdfGrid with styles from [PdfGridBuiltinStyle](#) Enum.

C#

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable.
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
//Add rows to the DataTable.
dataTable.Rows.Add(new object[] { "E01", "Clay" });
dataTable.Rows.Add(new object[] { "E02", "Thomas" });
dataTable.Rows.Add(new object[] { "E03", "George" });
dataTable.Rows.Add(new object[] { "E04", "Stefan" });
dataTable.Rows.Add(new object[] { "E05", "Mathew" });
//Assign data source.
pdfGrid.DataSource = dataTable;
//Apply built-in table style
pdfGrid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable4Accent1);
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new PointF(10, 10));
//Save the document.
doc.Save("Output.pdf");
//close the document
doc.Close(true);
```

VB.NET

```

'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page.
Dim page As PdfPage = doc.Pages.Add()
'Create a PdfGrid.
Dim pdfGrid As New PdfGrid()
'Create a DataTable.
Dim dataTable As New DataTable()
'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
'Add rows to the DataTable.
dataTable.Rows.Add(New Object() {"E01", "Clay"})
dataTable.Rows.Add(New Object() {"E02", "Thomas"})
dataTable.Rows.Add(New Object() {"E03", "George"})
dataTable.Rows.Add(New Object() {"E04", "Stefan"})
dataTable.Rows.Add(New Object() {"E05", "Mathew"})
'Assign data source.
pdfGrid.DataSource = dataTable
'Apply built-in table style
pdfGrid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable4Accent1)
'Draw grid to the page of PDF document.
pdfGrid.Draw(page, New PointF(10, 10))
'Save the document.
doc.Save("Output.pdf")
'close the document
doc.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E03", Name = "George" };
Object row4 = new { ID = "E04", Name = "Steffen" };
Object row5 = new { ID = "E05", Name = "Mathew" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
data.Add(row4);
data.Add(row5);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Apply built-in table style
pdfGrid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable4Accent1);

```

```
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new PointF(10, 10));
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E03", Name = "George" };
Object row4 = new { ID = "E04", Name = "Steffen" };
Object row5 = new { ID = "E05", Name = "Mathew" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
data.Add(row4);
data.Add(row5);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Apply built-in table style
pdfGrid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable4Accent1);
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new Syncfusion.Drawing.PointF(10, 10));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

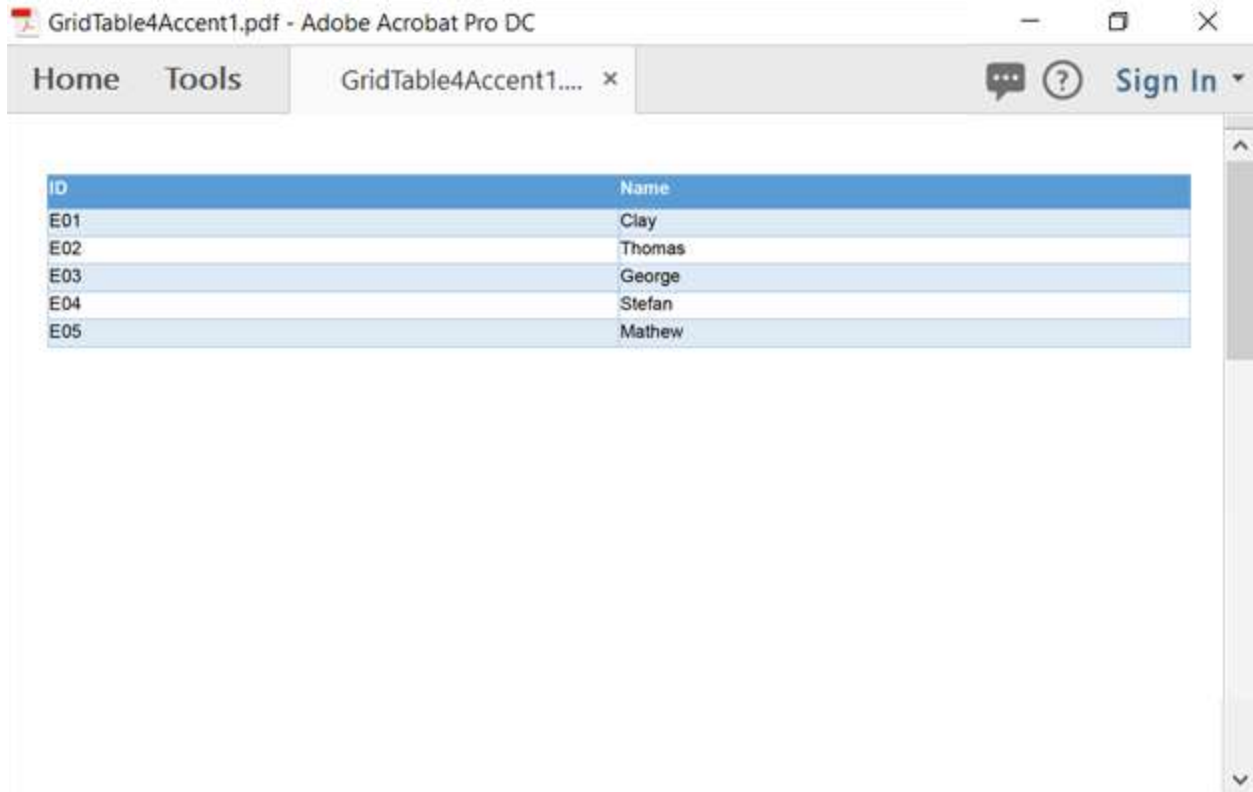
XAMARIN

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E03", Name = "George" };
Object row4 = new { ID = "E04", Name = "Steffen" };
Object row5 = new { ID = "E05", Name = "Mathew" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
data.Add(row4);
data.Add(row5);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Apply built-in table style
pdfGrid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable4Accent1);
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new Syncfusion.Drawing.PointF(10, 10));
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following image shows the PDF document with PdfGridBuiltinStyle.GridTable4Accent1.



The below code example illustrates how to apply built-in table style using [ApplyBuiltinStyle](#) method of the PdfLightTable with styles from [PdfLightTableBuiltinStyle](#) Enum.

C#

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Create a DataTable.
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
//Add rows to the DataTable.
dataTable.Rows.Add(new object[] { "E01", "Clay" });
dataTable.Rows.Add(new object[] { "E02", "Thomas" });
dataTable.Rows.Add(new object[] { "E03", "George" });
dataTable.Rows.Add(new object[] { "E04", "Stefan" });
dataTable.Rows.Add(new object[] { "E05", "Mathew" });
//Assign data source.
pdfLightTable.DataSource = dataTable;
//Apply built-in table style
pdfLightTable.ApplyBuiltinStyle(PdfLightTableBuiltinStyle.GridTable4Accent2);
;
//Draw grid to the page of PDF document.
pdfLightTable.Draw(page, new PointF(10, 10));
//Save the document.
```

```
doc.Save("Output.pdf");
//close the document
doc.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page.
Dim page As PdfPage = doc.Pages.Add()
'Create a PdfLightTable.
Dim pdfLightTable As New PdfLightTable()
'Create a DataTable.
Dim dataTable As New DataTable()
'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
'Add rows to the DataTable.
dataTable.Rows.Add(New Object() {"E01", "Clay"})
dataTable.Rows.Add(New Object() {"E02", "Thomas"})
dataTable.Rows.Add(New Object() {"E03", "George"})
dataTable.Rows.Add(New Object() {"E04", "Stefan"})
dataTable.Rows.Add(New Object() {"E05", "Mathew"})
'Assign data source.
pdfLightTable.DataSource = dataTable
'Apply built-in table style
pdfLightTable.ApplyBuiltinStyle(PdfLightTableBuiltinStyle.GridTable4Accent2)
'Draw grid to the page of PDF document.
pdfLightTable.Draw(page, New PointF(10, 10))
'Save the document.
doc.Save("Output.pdf")
'close the document
doc.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E03", Name = "George" };
Object row4 = new { ID = "E04", Name = "Steffen" };
Object row5 = new { ID = "E05", Name = "Mathew" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
data.Add(row4);
data.Add(row5);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
```



```
//Assign data source.
pdfLightTable.DataSource = dataTable;
//Apply built-in table style
pdfLightTable.ApplyBuiltinStyle(PdfLightTableBuiltinStyle.GridTable4Accent2)
;
//Draw grid to the page of PDF document.
pdfLightTable.Draw(page, new PointF(10, 10));
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

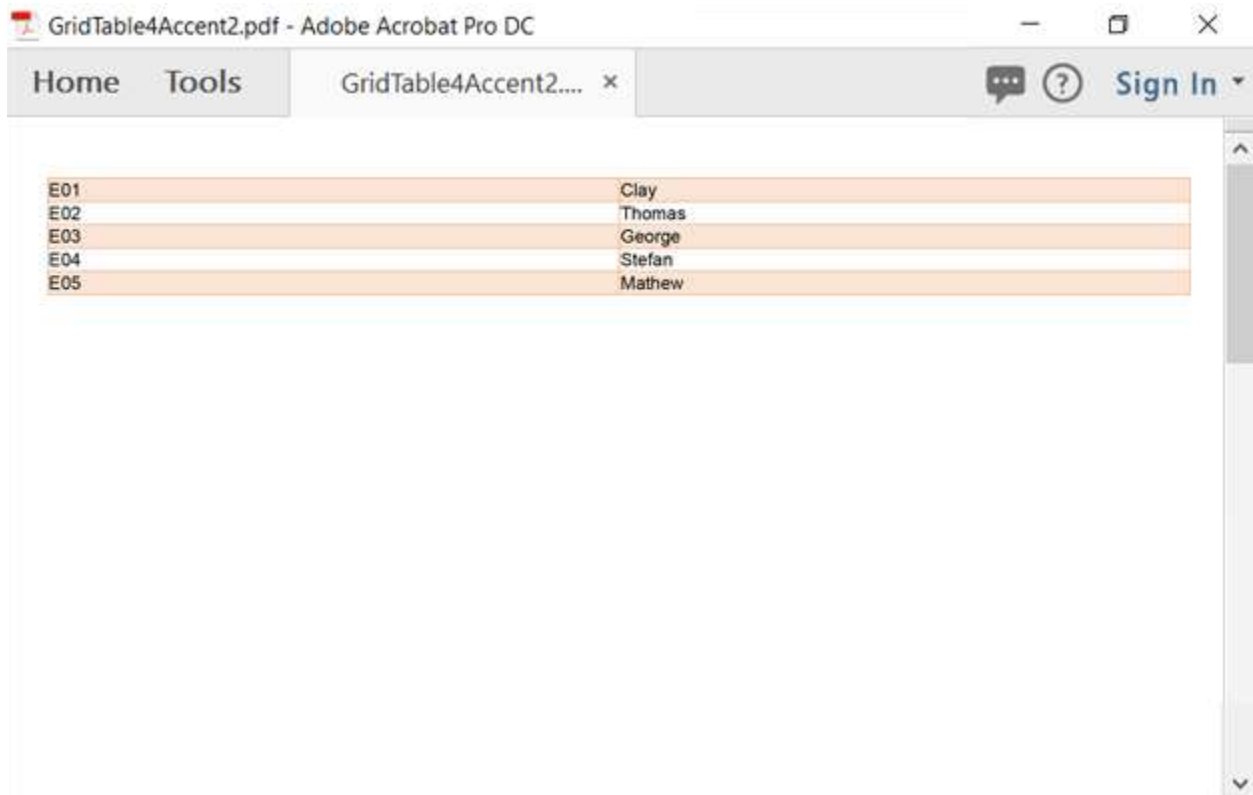
```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E03", Name = "George" };
Object row4 = new { ID = "E04", Name = "Steffen" };
Object row5 = new { ID = "E05", Name = "Mathew" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
data.Add(row4);
data.Add(row5);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfLightTable.DataSource = dataTable;
//Apply built-in table style
pdfLightTable.ApplyBuiltinStyle(PdfLightTableBuiltinStyle.GridTable4Accent2)
;
//Draw grid to the page of PDF document.
pdfLightTable.Draw(page, new Syncfusion.Drawing.PointF(10, 10));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
```

```
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E03", Name = "George" };
Object row4 = new { ID = "E04", Name = "Steffen" };
Object row5 = new { ID = "E05", Name = "Mathew" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
data.Add(row4);
data.Add(row5);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfLightTable.DataSource = dataTable;
//Apply built-in table style
pdfLightTable.ApplyBuiltinStyle(PdfLightTableBuiltinStyle.GridTable4Accent2);
;
//Draw grid to the page of PDF document.
pdfLightTable.Draw(page, new Syncfusion.Drawing.PointF(10, 10));
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

The following image shows the PDF document with PdfGridBuiltinStyle.Gridtable4Accent2.



The below code example illustrates how to apply built-in table styles with table options to the [PdfGrid](#).

C#

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable.
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
//Add rows to the DataTable.
dataTable.Rows.Add(new object[] { "E01", "Clay" });
dataTable.Rows.Add(new object[] { "E02", "Thomas" });
dataTable.Rows.Add(new object[] { "E03", "George" });
dataTable.Rows.Add(new object[] { "E04", "Stefan" });
dataTable.Rows.Add(new object[] { "E05", "Mathew" });
//Assign data source.
pdfGrid.DataSource = dataTable;
PdfGridBuiltinStyleSettings tableStyleOption = new
PdfGridBuiltinStyleSettings();
tableStyleOption.ApplyStyleForBandedRows = true;
tableStyleOption.ApplyStyleForHeaderRow = true;
//Apply built-in table style
```

```
pdfGrid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable4Accent4,
tableStyleOption);
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new PointF(10, 10));
//Save the document.
doc.Save("Output.pdf");
//close the document
doc.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page.
Dim page As PdfPage = doc.Pages.Add()
'Create a PdfGrid.
Dim pdfGrid As New PdfGrid()
'Create a DataTable.
Dim dataTable As New DataTable()
'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
'Add rows to the DataTable.
dataTable.Rows.Add(New Object() { "E01", "Clay" })
dataTable.Rows.Add(New Object() { "E02", "Thomas" })
dataTable.Rows.Add(New Object() { "E03", "George" })
dataTable.Rows.Add(New Object() { "E04", "Stefan" })
dataTable.Rows.Add(New Object() { "E05", "Mathew" })
'Assign data source.
pdfGrid.DataSource = dataTable
Dim tableStyleOption As New PdfGridBuiltinStyleSettings()
tableStyleOption.ApplyStyleForBandedRows = True
tableStyleOption.ApplyStyleForHeaderRow = True
'Apply built-in table style
pdfGrid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable4Accent4,
tableStyleOption)
'Draw grid to the page of PDF document.
pdfGrid.Draw(page, New PointF(10, 10))
'Save the document.
doc.Save("Output.pdf")
'close the document
doc.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E03", Name = "George" };
```

```

Object row4 = new { ID = "E04", Name = "Steffen" };
Object row5 = new { ID = "E05", Name = "Mathew" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
data.Add(row4);
data.Add(row5);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
PdfGridBuiltinStyleSettings tableStyleOption = new
PdfGridBuiltinStyleSettings();
tableStyleOption.ApplyStyleForBandedRows = true;
tableStyleOption.ApplyStyleForHeaderRow = true;
//Apply built-in table style
pdfGrid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable4Accent4,
tableStyleOption);
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new PointF(10, 10));
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E03", Name = "George" };
Object row4 = new { ID = "E04", Name = "Steffen" };
Object row5 = new { ID = "E05", Name = "Mathew" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
data.Add(row4);
data.Add(row5);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
PdfGridBuiltinStyleSettings tableStyleOption = new
PdfGridBuiltinStyleSettings();
tableStyleOption.ApplyStyleForBandedRows = true;

```

```

tableStyleOption.ApplyStyleForHeaderRow = true;
//Apply built-in table style
pdfGrid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable4Accent4,
tableStyleOption);
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new Syncfusion.Drawing.PointF(10, 10));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
doc.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page.
PdfPage page = doc.Pages.Add();
//Create a PdfGrid.
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
Object row3 = new { ID = "E03", Name = "George" };
Object row4 = new { ID = "E04", Name = "Steffen" };
Object row5 = new { ID = "E05", Name = "Mathew" };
data.Add(row1);
data.Add(row2);
data.Add(row3);
data.Add(row4);
data.Add(row5);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
PdfGridBuiltinStyleSettings tableStyleOption = new
PdfGridBuiltinStyleSettings();
tableStyleOption.ApplyStyleForBandedRows = true;
tableStyleOption.ApplyStyleForHeaderRow = true;
//Apply built-in table style
pdfGrid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable4Accent4,
tableStyleOption);
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new Syncfusion.Drawing.PointF(10, 10));
//Save the PDF document to stream.

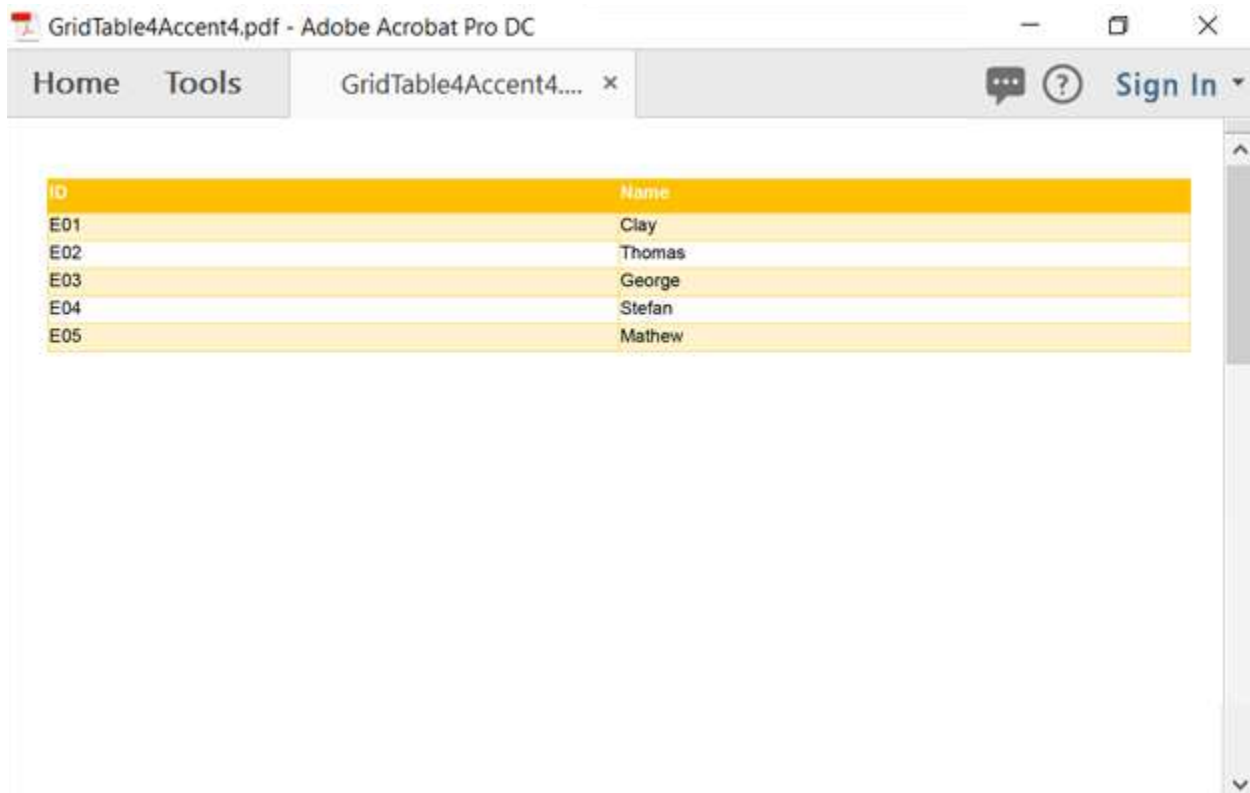
```

```

MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following image shows the PDF document with PdfGridBuiltinStyle.Gridtable4Accent4.



Pagination

Pagination in PdfLightTable

Essential PDF provides support to paginate the PdfLightTable using PdfLightTableLayoutFormat class.

The below sample illustrates how to allow the PdfLightTable to flow across pages.

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
// Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
// Initialize DataTable to assign as Data Source to the light table.
DataTable table = new DataTable();
//Include columns to the Data Table.
table.Columns.Add("Name");
table.Columns.Add("Age");
table.Columns.Add("Sex");
//Include rows to the Data Table.//you can add multiple rows
table.Rows.Add(new string[] { "abc", "21", "Male" });
//Assign data source.
pdfLightTable.DataSource = table;
//Set properties to paginate the table.
PdfLightTableLayoutFormat layoutFormat = new PdfLightTableLayoutFormat();
layoutFormat.Break = PdfLayoutBreakType.FitPage;
layoutFormat.Layout = PdfLayoutType.Paginate;
//Draw PdfLightTable.
pdfLightTable.Draw(page, new PointF(0, 0), layoutFormat);
//Save the document.
document.Save("Output.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
'Add a page.
Dim page As PdfPage = document.Pages.Add()
' Create a PdfLightTable.
Dim pdfLightTable As New PdfLightTable()
' Initialize DataTable to assign as Data Source to the light table.
Dim table As New DataTable()
'Include columns to the Data Table.
table.Columns.Add("Name")
table.Columns.Add("Age")
table.Columns.Add("Sex")
'Include rows to the Data Table.//you can add multiple rows
table.Rows.Add(New String() {"abc", "21", "Male"})
'Assign data source.
pdfLightTable.DataSource = table
'Set properties to paginate the table.
Dim layoutFormat As New PdfLightTableLayoutFormat()
layoutFormat.Break = PdfLayoutBreakType.FitPage
layoutFormat.Layout = PdfLayoutType.Paginate
'Draw PdfLightTable.
pdfLightTable.Draw(page, New PointF(0, 0), layoutFormat)
'Save the document.
document.Save("Output.pdf")
'Close the document
document.Close(True)

```


UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
// Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
//you can add multiple rows
Object row = new { Name = "abc", Age = "21", Sex = "Male" };
data.Add(row);
//Add list to IEnumerable
IEnumerable<object> table = data;
//Assign data source.
pdfLightTable.DataSource = table;
//Set properties to paginate the table.
PdfLightTableLayoutFormat layoutFormat = new PdfLightTableLayoutFormat();
layoutFormat.Break = PdfLayoutBreakType.FitPage;
layoutFormat.Layout = PdfLayoutType.Paginate;
//Draw PdfLightTable.
pdfLightTable.Draw(page, new PointF(0, 0), layoutFormat);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
// Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
//you can add multiple rows
Object row = new { Name = "abc", Age = "21", Sex = "Male" };
data.Add(row);
//Add list to IEnumerable
IEnumerable<object> table = data;
//Assign data source.
pdfLightTable.DataSource = table;
//Set properties to paginate the table.
PdfLightTableLayoutFormat layoutFormat = new PdfLightTableLayoutFormat();
layoutFormat.Break = PdfLayoutBreakType.FitPage;
layoutFormat.Layout = PdfLayoutType.Paginate;
//Draw PdfLightTable.
pdfLightTable.Draw(page, new Syncfusion.Drawing.PointF(0, 0), layoutFormat);
//Creating the stream object
```

```

MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
// Create a PdfLightTable.
PdfLightTable pdfLightTable = new PdfLightTable();
//Add values to list
List<object> data = new List<object>();
//you can add multiple rows
Object row = new { Name = "abc", Age = "21", Sex = "Male" };
data.Add(row);
//Add list to IEnumerable
IEnumerable<object> table = data;
//Assign data source.
pdfLightTable.DataSource = table;
//Set properties to paginate the table.
PdfLightTableLayoutFormat layoutFormat = new PdfLightTableLayoutFormat();
layoutFormat.Break = PdfLayoutBreakType.FitPage;
layoutFormat.Layout = PdfLayoutType.Paginate;
//Draw PdfLightTable.
pdfLightTable.Draw(page, new Syncfusion.Drawing.PointF(0, 0), layoutFormat);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{

```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Pagination in PdfGrid

Essential PDF supports to paginate the [PdfGrid](#) using [PdfGridLayoutFormat](#) class.

The below sample illustrates how to allow the PdfGrid to flow across pages.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "E01", "Clay" });
dataTable.Rows.Add(new object[] { "E02", "Thomas" });
//Assign data source.
pdfGrid.DataSource = dataTable;
//Set properties to paginate the grid.
PdfGridLayoutFormat layoutFormat = new PdfGridLayoutFormat();
layoutFormat.Break = PdfLayoutBreakType.FitPage;
layoutFormat.Layout = PdfLayoutType.Paginate;
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new PointF(10, 10), layoutFormat);
//Save the document.
document.Save("Output.pdf");
//close the document
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Add a page.
Dim page As PdfPage = document.Pages.Add()
'Create a PdfGrid.
Dim pdfGrid As New PdfGrid()
'Create a DataTable.
Dim dataTable As New DataTable()
'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
'Add rows to the DataTable
dataTable.Rows.Add(New Object() { "E01", "Clay" })
dataTable.Rows.Add(New Object() { "E02", "Thomas" })
'Assign data source.
pdfGrid.DataSource = dataTable
```

```

'Set properties to paginate the grid
Dim layoutFormat As New PdfGridLayoutFormat()
layoutFormat.Break = PdfLayoutBreakType.FitPage
layoutFormat.Layout = PdfLayoutType.Paginate
'Draw grid to the page of PDF document.
pdfGrid.Draw(page, New PointF(10, 10), layoutFormat)
'Save the document
document.Save("Output.pdf")
'close the document
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
//You can add multiple rows
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
data.Add(row1);
data.Add(row2);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Set properties to paginate the grid.
PdfGridLayoutFormat layoutFormat = new PdfGridLayoutFormat();
layoutFormat.Break = PdfLayoutBreakType.FitPage;
layoutFormat.Layout = PdfLayoutType.Paginate;
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new PointF(10, 10), layoutFormat);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
//You can add multiple rows here

```

```

Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
data.Add(row1);
data.Add(row2);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Set properties to paginate the grid.
PdfGridLayoutFormat layoutFormat = new PdfGridLayoutFormat();
layoutFormat.Break = PdfLayoutBreakType.FitPage;
layoutFormat.Layout = PdfLayoutType.Paginate;
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new Syncfusion.Drawing.PointF(10, 10), layoutFormat);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
//You can add multiple rows
Object row1 = new { ID = "E01", Name = "Clay" };
Object row2 = new { ID = "E02", Name = "Thomas" };
data.Add(row1);
data.Add(row2);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source.
pdfGrid.DataSource = dataTable;
//Set properties to paginate the grid.
PdfGridLayoutFormat layoutFormat = new PdfGridLayoutFormat();
layoutFormat.Break = PdfLayoutBreakType.FitPage;
layoutFormat.Layout = PdfLayoutType.Paginate;
//Draw grid to the page of PDF document.
pdfGrid.Draw(page, new Syncfusion.Drawing.PointF(10, 10), layoutFormat);
//Save the PDF document to stream.

```

```

MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Adjust table width automatically

You can automatically adjust the width of the table by enabling the [AllowHorizontalOverflow](#) property of [PdfGridStyle](#) instance. The following code snippet illustrates this.

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add new section to the document
PdfSection section = document.Sections.Add();
//Add a page to the section
PdfPage page = section.Pages.Add();
//Initialize PdfGrid
PdfGrid grid = new PdfGrid();
//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("Employee_ID");
dataTable.Columns.Add("Employee_Name");
dataTable.Columns.Add("Employee_Role");
dataTable.Columns.Add("Employee_DateOfBirth");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "E01", "Clay", "Sales Representative",
"12/8/1948" });
dataTable.Rows.Add(new object[] { "E02", "Thomas", "Sales Representative",
"7/2/1963" });
dataTable.Rows.Add(new object[] { "E03", "Ash", "Sales Manager", "3/4/1955"
});
dataTable.Rows.Add(new object[] { "E04", "Andrew", "Vice President, Sales",
"2/19/1952" });
//Assign data source to grid
grid.DataSource = dataTable;
//Apply the table style
grid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable5DarkAccent5);
//Allow the horizontal overflow for PdfGrid
grid.Style.AllowHorizontalOverflow = true;

```

```
//Draw the PdfGrid on page
grid.Draw(page, PointF.Empty);
//Save the PDF document
document.Save("Output.pdf");
//Close the instance of PdfDocument
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add new section to the document
Dim section As PdfSection = document.Sections.Add
'Add a page to the section
Dim page As PdfPage = section.Pages.Add
'Initialize PdfGrid
Dim grid As PdfGrid = New PdfGrid
'Create a DataTable
Dim dataTable As DataTable = New DataTable
'Add columns to the DataTable
dataTable.Columns.Add("Employee_ID")
dataTable.Columns.Add("Employee_Name")
dataTable.Columns.Add("Employee_Role")
dataTable.Columns.Add("Employee_DateOfBirth")
'Add rows to the DataTable
dataTable.Rows.Add(New Object() {"E01", "Clay", "Sales Representative",
"12/8/1948"})
dataTable.Rows.Add(New Object() {"E02", "Thomas", "Sales Representative",
"7/2/1963"})
dataTable.Rows.Add(New Object() {"E03", "Ash", "Sales Manager",
"3/4/1955"})
dataTable.Rows.Add(New Object() {"E04", "Andrew", "Vice President, Sales",
"2/19/1952"})
'Assign data source to grid
grid.DataSource = dataTable
'Apply the table style
grid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable5DarkAccent5)
'Allow the horizontal overflow for PdfGrid
grid.Style.AllowHorizontalOverflow = True
'Draw the PdfGrid on page
grid.Draw(page, PointF.Empty)
'Save the PDF document
document.Save("Output.pdf")
'Close the instance of PdfDocument
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add new section to the document
PdfSection section = document.Sections.Add();
//Add a page to the section
PdfPage page = section.Pages.Add();
//Initialize PdfGrid
PdfGrid grid = new PdfGrid();
```

```
//Add values to list
List<object> data = new List<object>();
Object gridrow1 = new { Employee_ID = "E01", Employee_Name = "Clay",
Employee_Role = "Sales Representative", Employee_DateOfBirth = "12/8/1948"
};
Object gridrow2 = new { Employee_ID = "E02", Employee_Name = "Thomas",
Employee_Role = "Sales Representative", Employee_DateOfBirth = "7/2/1963" };
Object gridrow3 = new { Employee_ID = "E03", Employee_Name = "Ash",
Employee_Role = "Sales Manager", Employee_DateOfBirth = "3/4/1955" };
Object gridrow4 = new { Employee_ID = "E04", Employee_Name = "Andrew",
Employee_Role = "Vice President, Sales", Employee_DateOfBirth = "2/19/1952"
};
data.Add(gridrow1);
data.Add(gridrow2);
data.Add(gridrow3);
data.Add(gridrow4);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source to grid
grid.DataSource = dataTable;
//Apply the table style
grid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable5DarkAccent5);
//Allow the horizontal overflow for PdfGrid
grid.Style.AllowHorizontalOverflow = true;
//Draw the PdfGrid on page
grid.Draw(page, PointF.Empty);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add new section to the document
PdfSection section = document.Sections.Add();
//Add a page to the section
PdfPage page = section.Pages.Add();
//Initialize PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridrow1 = new { Employee_ID = "E01", Employee_Name = "Clay",
Employee_Role = "Sales Representative", Employee_DateOfBirth = "12/8/1948"
};
Object gridrow2 = new { Employee_ID = "E02", Employee_Name = "Thomas",
Employee_Role = "Sales Representative", Employee_DateOfBirth = "7/2/1963" };
Object gridrow3 = new { Employee_ID = "E03", Employee_Name = "Ash",
Employee_Role = "Sales Manager", Employee_DateOfBirth = "3/4/1955" };
```



```

Object gridlrow4 = new { Employee_ID = "E04", Employee_Name = "Andrew",
Employee_Role = "Vice President, Sales", Employee_DateOfBirth = "2/19/1952"
};
data.Add(gridlrow1);
data.Add(gridlrow2);
data.Add(gridlrow3);
data.Add(gridlrow4);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source to grid
grid.DataSource = dataTable;
//Apply the table style
grid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable5DarkAccent5);
//Allow the horizontal overflow for PdfGrid
grid.Style.AllowHorizontalOverflow = true;
//Draw the PdfGrid on page
grid.Draw(page, PointF.Empty);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add new section to the document
PdfSection section = document.Sections.Add();
//Add a page to the section
PdfPage page = section.Pages.Add();
//Initialize PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridlrow1 = new { Employee_ID = "E01", Employee_Name = "Clay",
Employee_Role = "Sales Representative", Employee_DateOfBirth = "12/8/1948"
};
Object gridlrow2 = new { Employee_ID = "E02", Employee_Name = "Thomas",
Employee_Role = "Sales Representative", Employee_DateOfBirth = "7/2/1963" };
Object gridlrow3 = new { Employee_ID = "E03", Employee_Name = "Ash",
Employee_Role = "Sales Manager", Employee_DateOfBirth = "3/4/1955" };
Object gridlrow4 = new { Employee_ID = "E04", Employee_Name = "Andrew",
Employee_Role = "Vice President, Sales", Employee_DateOfBirth = "2/19/1952"
};
data.Add(gridlrow1);
data.Add(gridlrow2);
data.Add(gridlrow3);
data.Add(gridlrow4);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;

```

```

//Assign data source to grid
grid.DataSource = dataTable;
//Apply the table style
grid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable5DarkAccent5);
//Allow the horizontal overflow for PdfGrid
grid.Style.AllowHorizontalOverflow = true;
//Draw the PdfGrid on page
grid.Draw(page, PointF.Empty);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Adding multiple tables

The Essential PDF supports maintaining the position of a PDF grid drawn on PDF page using [PdfGridLayoutResult](#). It provides the rendered bounds of previously added grid, which can be used to place successive elements without overlapping. You can add multiple PDF grids using the bottom position of previously rendered PDF grid. The following code snippet illustrates this.

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create a new PdfGrid instance
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
dataTable.Columns.Add("Salary");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
dataTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
dataTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//Assign data source
pdfGrid.DataSource = dataTable;

```

```

//Draw grid on the page of PDF document and store the grid position in PdfGridLayoutResult
PdfGridLayoutResult pdfGridLayoutResult = pdfGrid.Draw(page, new PointF(10, 10));
//Initialize PdfGrid and DataTable
pdfGrid = new PdfGrid();
dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("Name");
dataTable.Columns.Add("Age");
dataTable.Columns.Add("Sex");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "Andrew", "21", "Male" });
dataTable.Rows.Add(new object[] { "Steven", "22", "Female" });
dataTable.Rows.Add(new object[] { "Michael", "24", "Male" });
//Assign data source
pdfGrid.DataSource = dataTable;
//Draw the grid on page using previous result
pdfGrid.Draw(page, new PointF(10, pdfGridLayoutResult.Bounds.Bottom + 20));
//Save the document
document.Save("Output.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add a page
Dim page As PdfPage = document.Pages.Add
'Create a new PdfGrid instance
Dim pdfGrid As PdfGrid = New PdfGrid
'Create a DataTable
Dim dataTable As DataTable = New DataTable
'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
dataTable.Columns.Add("Salary")
'Add rows to the DataTable
dataTable.Rows.Add(New Object() { "E01", "Clay", "$10,000" })
dataTable.Rows.Add(New Object() { "E02", "Thomas", "$10,500" })
dataTable.Rows.Add(New Object() { "E03", "Simon", "$12,000" })
'Assign data source
pdfGrid.DataSource = dataTable
'Draw grid on the page of PDF document and store the grid position in PdfGridLayoutResult
Dim pdfGridLayoutResult As PdfGridLayoutResult = pdfGrid.Draw(page, New PointF(10, 10))
'Initialize PdfGrid and DataTable
pdfGrid = New PdfGrid
dataTable = New DataTable
'Add columns to the DataTable
dataTable.Columns.Add("Name")
dataTable.Columns.Add("Age")
dataTable.Columns.Add("Sex")
'Add rows to the DataTable

```

```

dataTable.Rows.Add(New Object() {"Andrew", "21", "Male"})
dataTable.Rows.Add(New Object() {"Steven", "22", "Female"})
dataTable.Rows.Add(New Object() {"Michael", "24", "Male"})
'Assign data source
pdfGrid.DataSource = dataTable
'Draw the grid on page using previous result
pdfGrid.Draw(page, New PointF(10, (pdfGridLayoutResult.Bounds.Bottom + 20)))
'Save the document
document.Save("Output.pdf")
'Close the document
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create a new PdfGrid instance
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object grid1row1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object grid1row2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object grid1row3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(grid1row1);
data.Add(grid1row2);
data.Add(grid1row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
pdfGrid.DataSource = dataTable;
//Draw grid on the page of PDF document and store the grid position in
PdfGridLayoutResult
PdfGridLayoutResult pdfGridLayoutResult = pdfGrid.Draw(page, new PointF(10,
10));
//Initialize PdfGrid and list
pdfGrid = new PdfGrid();
data = new List<object>();
//Add values to list
Object grid2row1 = new { Name = "Andrew", Age = "21", Sex = "Male" };
Object grid2row2 = new { Name = "Steven", Age = "22", Sex = "Female" };
Object grid2row3 = new { Name = "Michael", Age = "24", Sex = "Male" };
data.Add(grid2row1);
data.Add(grid2row2);
data.Add(grid2row3);
//Add list to IEnumerable
dataTable = data;
//Assign data source
pdfGrid.DataSource = dataTable;
//Draw the grid on page using previously result
pdfGrid.Draw(page, new PointF(10, pdfGridLayoutResult.Bounds.Bottom + 20));
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
document.Save(stream);

```

```
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create a new PdfGrid instance
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object grid1row1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object grid1row2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object grid1row3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(grid1row1);
data.Add(grid1row2);
data.Add(grid1row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
pdfGrid.DataSource = dataTable;
//Draw grid on the page of PDF document and store the grid position in
PdfGridLayoutResult
PdfGridLayoutResult pdfGridLayoutResult = pdfGrid.Draw(page, new PointF(10,
10));
//Initialize PdfGrid and list
pdfGrid = new PdfGrid();
data = new List<object>();
//Add values to the list
Object grid2row1 = new { Name = "Andrew", Age = "21", Sex = "Male" };
Object grid2row2 = new { Name = "Steven", Age = "22", Sex = "Female" };
Object grid2row3 = new { Name = "Michael", Age = "24", Sex = "Male" };
data.Add(grid2row1);
data.Add(grid2row2);
data.Add(grid2row3);
//Add list to IEnumerable
dataTable = data;
//Assign data source
pdfGrid.DataSource = dataTable;
//Draw the grid on page using previous result
pdfGrid.Draw(page, new PointF(10, pdfGridLayoutResult.Bounds.Bottom + 20));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create a new PdfGrid instance
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object grid1row1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object grid1row2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object grid1row3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(grid1row1);
data.Add(grid1row2);
data.Add(grid1row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
pdfGrid.DataSource = dataTable;
//Draw grid on the page of PDF document and store the grid position in
PdfGridLayoutResult
PdfGridLayoutResult pdfGridLayoutResult = pdfGrid.Draw(page, new PointF(10,
10));
//Initialize PdfGrid and list
pdfGrid = new PdfGrid();
data = new List<object>();
//Add values to list
Object grid2row1 = new { Name = "Andrew", Age = "21", Sex = "Male" };
Object grid2row2 = new { Name = "Steven", Age = "22", Sex = "Female" };
Object grid2row3 = new { Name = "Michael", Age = "24", Sex = "Male" };
data.Add(grid2row1);
data.Add(grid2row2);
data.Add(grid2row3);
//Add list to IEnumerable
dataTable = data;
//Assign data source
pdfGrid.DataSource = dataTable;
//Draw the grid on page using previous result
pdfGrid.Draw(page, new PointF(10, pdfGridLayoutResult.Bounds.Bottom + 20));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}

```

```
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

String formatting

Essential PDF supports applying string formatting for whole table, a column in table, a row in table and a cell in table using the [PdfStringFormat](#) class.

String formatting for whole table in PdfGrid

The following code snippet explains how to apply string formatting for whole table in [PdfGrid](#).

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
dataTable.Columns.Add("Salary");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
dataTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
dataTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//Assign data source
grid.DataSource = dataTable;
//Create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string formatting for whole table
for (int i = 0; i < grid.Columns.Count; i++)
{
grid.Columns[i].Format = stringFormat;
}
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Save the document and close the instance of PdfDocument
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add page to the document
Dim page As PdfPage = document.Pages.Add
```

```

'Create a new PdfGrid
Dim grid As PdfGrid = New PdfGrid
'Create a DataTable
Dim dataTable As DataTable = New DataTable
'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
dataTable.Columns.Add("Salary")
'Add rows to the DataTable
dataTable.Rows.Add(New Object() {"E01", "Clay", "$10,000"})
dataTable.Rows.Add(New Object() {"E02", "Thomas", "$10,500"})
dataTable.Rows.Add(New Object() {"E03", "Simon", "$12,000"})
'Assign data source
grid.DataSource = dataTable
'Create and customize the string formats
Dim stringFormat As PdfStringFormat = New PdfStringFormat
stringFormat.Alignment = PdfTextAlignment.Center
stringFormat.LineAlignment = PdfVerticalAlignment.Middle
stringFormat.CharacterSpacing = 2.0F
'Apply string formatting for whole table
Dim i As Integer = 0
For i = 0 To grid.Columns.Count - 1 Step 1
    grid.Columns(i).Format = stringFormat
Next
'Draw the PdfGrid on page
grid.Draw(page, New PointF(10, 10))
'Save the document and close the instance of PdfDocument
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object grid1row1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object grid1row2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object grid1row3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(grid1row1);
data.Add(grid1row2);
data.Add(grid1row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
grid.DataSource = dataTable;
//Create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string formatting for whole table

```



```

for (int i = 0; i < grid.Columns.Count; i++)
{
    grid.Columns[i].Format = stringFormat;
}
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Create memory stream
MemoryStream ms = new MemoryStream();
//Open the document in browser after saving it
document.Save(ms);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respective code samples
Save(ms, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridrow1);
data.Add(gridrow2);
data.Add(gridrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
grid.DataSource = dataTable;
//Create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string formatting for whole table
for (int i = 0; i < grid.Columns.Count; i++)
{
    grid.Columns[i].Format = stringFormat;
}
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");

```

```
fileStreamResult.FileDownloadName = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridlrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridlrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridlrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridlrow1);
data.Add(gridlrow2);
data.Add(gridlrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
grid.DataSource = dataTable;
//Create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string formatting for whole table
for (int i = 0; i < grid.Columns.Count; i++)
{
    grid.Columns[i].Format = stringFormat;
}
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

String formatting for whole table in PdfLightTable

The following code snippet explains how to add string formatting for whole table in [PdfLightTable](#).

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a PdfLightTable
PdfLightTable lightTable = new PdfLightTable();
//Set the DataSourceType as Direct
lightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns
lightTable.Columns.Add(new PdfColumn("ID"));
lightTable.Columns.Add(new PdfColumn("Name"));
lightTable.Columns.Add(new PdfColumn("Salary"));
//Add rows
lightTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
lightTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
lightTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//Enable ShowHeader
lightTable.Style.ShowHeader = true;
//Create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string formatting for whole table
for (int i = 0; i < lightTable.Columns.Count; i++)
{
    lightTable.Columns[i].StringFormat = stringFormat;
}
//Draw the PdfLightTable on page
lightTable.Draw(page, new PointF(10, 10));
//Save the document and close the instance of PdfDocument
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add page to the document
Dim page As PdfPage = document.Pages.Add
'Create a PdfLightTable
Dim lightTable As PdfLightTable = New PdfLightTable
'Set the DataSourceType as Direct
lightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect
'Create columns
lightTable.Columns.Add(New PdfColumn("ID"))
lightTable.Columns.Add(New PdfColumn("Name"))
lightTable.Columns.Add(New PdfColumn("Salary"))
'Add rows
lightTable.Rows.Add(New Object() { "E01", "Clay", "$10,000" })
lightTable.Rows.Add(New Object() { "E02", "Thomas", "$10,500" })
```

```

lightTable.Rows.Add(New Object() {"E03", "Simon", "$12,000"})
'Enable ShowHeader
lightTable.Style.ShowHeader = True
'Create and customize the string formats
Dim stringFormat As PdfStringFormat = New PdfStringFormat
stringFormat.Alignment = PdfTextAlignment.Center
stringFormat.LineAlignment = PdfVerticalAlignment.Middle
stringFormat.CharacterSpacing = 2.0F
'Apply string formatting for whole table
Dim i As Integer = 0
For i = 0 To lightTable.Columns.Count - 1 Step 1
lightTable.Columns(i).StringFormat = stringFormat
Next
'Draw the PdfLightTable on page
lightTable.Draw(page, New PointF(10, 10))
'Save the document and close the instance of PdfDocument
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a PdfLightTable
PdfLightTable lightTable = new PdfLightTable();
//Set the DataSourceType as Direct
lightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns
lightTable.Columns.Add(new PdfColumn("ID"));
lightTable.Columns.Add(new PdfColumn("Name"));
lightTable.Columns.Add(new PdfColumn("Salary"));
//Add rows
lightTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
lightTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
lightTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//Enable ShowHeader
lightTable.Style.ShowHeader = true;
//Create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string formatting for whole table
for (int i = 0; i < lightTable.Columns.Count; i++)
{
lightTable.Columns[i].StringFormat = stringFormat;
}
//Draw the PdfLightTable on page
lightTable.Draw(page, new PointF(10, 10));
//Create memory stream
MemoryStream ms = new MemoryStream();
//Open the document in browser after saving it
document.Save(ms);
//Close the document

```

```
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respective code samples
Save(ms, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a PdfLightTable
PdfLightTable lightTable = new PdfLightTable();
//Set the DataSourceType as Direct
lightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns
lightTable.Columns.Add(new PdfColumn("ID"));
lightTable.Columns.Add(new PdfColumn("Name"));
lightTable.Columns.Add(new PdfColumn("Salary"));
//Add rows
lightTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
lightTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
lightTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//Enable ShowHeader
lightTable.Style.ShowHeader = true;
//Create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string formatting for whole table
for (int i = 0; i < lightTable.Columns.Count; i++)
{
    lightTable.Columns[i].StringFormat = stringFormat;
}
//Draw the PdfLightTable on page
lightTable.Draw(page, new PointF(10, 10));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a PdfLightTable
PdfLightTable lightTable = new PdfLightTable();
```

```

//Set the DataSourceType as Direct
lightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns
lightTable.Columns.Add(new PdfColumn("ID"));
lightTable.Columns.Add(new PdfColumn("Name"));
lightTable.Columns.Add(new PdfColumn("Salary"));
//Add rows
lightTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
lightTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
lightTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//Enable ShowHeader
lightTable.Style.ShowHeader = true;
//Create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string formatting for whole table
for (int i = 0; i < lightTable.Columns.Count; i++)
{
    lightTable.Columns[i].StringFormat = stringFormat;
}
//Draw the PdfLightTable on page
lightTable.Draw(page, new PointF(10, 10));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

String formatting to a column in PdfGrid

The following code snippet explains how to add string formatting to a column in [PdfGrid](#).

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();

```

```

//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
dataTable.Columns.Add("Salary");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
dataTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
dataTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//Assign data source
grid.DataSource = dataTable;
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string formatting to a column
grid.Columns[1].Format = stringFormat;
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Save the document and close the instance of PdfDocument
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add page to the document
Dim page As PdfPage = document.Pages.Add
'Create a new PdfGrid
Dim grid As PdfGrid = New PdfGrid
'Create a DataTable
Dim dataTable As DataTable = New DataTable
'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
dataTable.Columns.Add("Salary")
'Add rows to the DataTable
dataTable.Rows.Add(New Object() {"E01", "Clay", "$10,000"})
dataTable.Rows.Add(New Object() {"E02", "Thomas", "$10,500"})
dataTable.Rows.Add(New Object() {"E03", "Simon", "$12,000"})
'Assign data source
grid.DataSource = dataTable
'Create and customize the string formats
Dim stringFormat As PdfStringFormat = New PdfStringFormat
stringFormat.Alignment = PdfTextAlignment.Center
stringFormat.LineAlignment = PdfVerticalAlignment.Middle
stringFormat.CharacterSpacing = 2.0F
'Apply string formatting to a column
grid.Columns(1).Format = stringFormat
'Draw the PdfGrid on page
grid.Draw(page, New PointF(10, 10))
'Save the document and close the instance of PdfDocument
document.Save("Output.pdf")

```

```
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridrow1);
data.Add(gridrow2);
data.Add(gridrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
grid.DataSource = dataTable;
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string formatting to a column
grid.Columns[1].Format = stringFormat;
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Create memory stream
MemoryStream ms = new MemoryStream();
//Open the document in browser after saving it
document.Save(ms);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respective code samples
Save(ms, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridrow1);
data.Add(gridrow2);
```



```

data.Add(grid1row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
grid.DataSource = dataTable;
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string formatting to a column
grid.Columns[1].Format = stringFormat;
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object grid1row1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object grid1row2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object grid1row3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(grid1row1);
data.Add(grid1row2);
data.Add(grid1row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
grid.DataSource = dataTable;
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string formatting to a column
grid.Columns[1].Format = stringFormat;
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Save the document to the stream
MemoryStream stream = new MemoryStream();

```

```

document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

String formatting to a column in PdfLightTable

The following code snippet explains how to add string formatting to a column in [PdfLightTable](#).

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a PdfLightTable
PdfLightTable lightTable = new PdfLightTable();
//Set the DataSourceType as Direct
lightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns
lightTable.Columns.Add(new PdfColumn("ID"));
lightTable.Columns.Add(new PdfColumn("Name"));
lightTable.Columns.Add(new PdfColumn("Salary"));
//Add rows
lightTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
lightTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
lightTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Enable ShowHeader
lightTable.Style.ShowHeader = true;
//Apply string format to a column
lightTable.Columns[1].StringFormat = stringFormat;
//Draw PdfLightTable on page
lightTable.Draw(page, new PointF(10, 10));
//Save the document and close the instance of PdfDocument
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add page to the document
Dim page As PdfPage = document.Pages.Add
'Create a PdfLightTable
Dim lightTable As PdfLightTable = New PdfLightTable
'Set the DataSourceType as Direct
lightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect
'Create columns
lightTable.Columns.Add(New PdfColumn("ID"))
lightTable.Columns.Add(New PdfColumn("Name"))
lightTable.Columns.Add(New PdfColumn("Salary"))
'Add rows
lightTable.Rows.Add(New Object() {"E01", "Clay", "$10,000"})
lightTable.Rows.Add(New Object() {"E02", "Thomas", "$10,500"})
lightTable.Rows.Add(New Object() {"E03", "Simon", "$12,000"})
'create and customize the string formats
Dim stringFormat As PdfStringFormat = New PdfStringFormat
stringFormat.Alignment = PdfTextAlignment.Center
stringFormat.LineAlignment = PdfVerticalAlignment.Middle
stringFormat.CharacterSpacing = 2.0F
'Enable ShowHeader
lightTable.Style.ShowHeader = True
'Apply string format to a column
lightTable.Columns(1).StringFormat = stringFormat
'Draw PdfLightTable on page
lightTable.Draw(page, New PointF(10, 10))
'Save the document and close the instance of PdfDocument
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a PdfLightTable
PdfLightTable lightTable = new PdfLightTable();
//Set the DataSourceType as Direct
lightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns
lightTable.Columns.Add(new PdfColumn("ID"));
lightTable.Columns.Add(new PdfColumn("Name"));
lightTable.Columns.Add(new PdfColumn("Salary"));
//Add rows
lightTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
lightTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
lightTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;

```

```

//Enable ShowHeader
lightTable.Style.ShowHeader = true;
//Apply string format to a column
lightTable.Columns[1].StringFormat = stringFormat;
//Draw PdfLightTable on page
lightTable.Draw(page, new PointF(10, 10));
//Create memory stream
MemoryStream ms = new MemoryStream();
//Open the document in browser after saving it
document.Save(ms);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respective code samples
Save(ms, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a PdfLightTable
PdfLightTable lightTable = new PdfLightTable();
//Set the DataSourceType as Direct
lightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns
lightTable.Columns.Add(new PdfColumn("ID"));
lightTable.Columns.Add(new PdfColumn("Name"));
lightTable.Columns.Add(new PdfColumn("Salary"));
//Add rows
lightTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
lightTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
lightTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Enable ShowHeader
lightTable.Style.ShowHeader = true;
//Apply string format to a column
lightTable.Columns[1].StringFormat = stringFormat;
//Draw PdfLightTable on page
lightTable.Draw(page, new PointF(10, 10));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownloadName = "Output.pdf";
return fileStreamResult;

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a PdfLightTable
PdfLightTable lightTable = new PdfLightTable();
//Set the DataSourceType as Direct
lightTable.DataSourceType = PdfLightTableDataSourceType.TableDirect;
//Create columns
lightTable.Columns.Add(new PdfColumn("ID"));
lightTable.Columns.Add(new PdfColumn("Name"));
lightTable.Columns.Add(new PdfColumn("Salary"));
//Add rows
lightTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
lightTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
lightTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Enable ShowHeader
lightTable.Style.ShowHeader = true;
//Apply string format to a column
lightTable.Columns[1].StringFormat = stringFormat;
//Draw PdfLightTable on page
lightTable.Draw(page, new PointF(10, 10));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

String formatting for a cell in PdfGrid

The following code snippet illustrates how to add string formatting for a cell in [PdfGrid](#).

C#

```

//Create a new PDF document

```

```

PdfDocument document = new PdfDocument();
//Add page to the PdfDocument
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
dataTable.Columns.Add("Salary");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
dataTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
dataTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//Assign data source
grid.DataSource = dataTable;
//Create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string format to a cell
grid.Rows[2].Cells[1].StringFormat = stringFormat;
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Save the document and close the instance of PdfDocument
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add page to the PdfDocument
Dim page As PdfPage = document.Pages.Add
'Create a new PdfGrid
Dim grid As PdfGrid = New PdfGrid
'Create a DataTable
Dim dataTable As DataTable = New DataTable
'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
dataTable.Columns.Add("Salary")
'Add rows to the DataTable
dataTable.Rows.Add(New Object() { "E01", "Clay", "$10,000" })
dataTable.Rows.Add(New Object() { "E02", "Thomas", "$10,500" })
dataTable.Rows.Add(New Object() { "E03", "Simon", "$12,000" })
'Assign data source
grid.DataSource = dataTable
'Create and customize the string formats
Dim stringFormat As PdfStringFormat = New PdfStringFormat
stringFormat.Alignment = PdfTextAlignment.Center
stringFormat.LineAlignment = PdfVerticalAlignment.Middle
stringFormat.CharacterSpacing = 2.0F
'Apply string format to a cell

```

```

grid.Rows(2).Cells(1).StringFormat = stringFormat
'Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10))
'Save the document and close the instance of PdfDocument
document.Save("Output.pdf")
document.Close(true)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the PdfDocument
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridrow1);
data.Add(gridrow2);
data.Add(gridrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
grid.DataSource = dataTable;
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string format to a cell
grid.Rows[2].Cells[1].StringFormat = stringFormat;
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Create memory stream
MemoryStream ms = new MemoryStream();
//Open the document in browser after saving it
document.Save(ms);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respective code samples
Save(ms, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the PdfDocument
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();

```

```

Object gridrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridrow1);
data.Add(gridrow2);
data.Add(gridrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
grid.DataSource = dataTable;
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string format to a cell
grid.Rows[2].Cells[1].StringFormat = stringFormat;
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the PdfDocument
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridrow1);
data.Add(gridrow2);
data.Add(gridrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
grid.DataSource = dataTable;
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Apply string format to a cell

```



```

grid.Rows[2].Cells[1].StringFormat = stringFormat;
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

String formatting for a row in PdfGrid

The following code snippet illustrates how to add string formatting for a row in [PdfGrid](#).

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the document
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
dataTable.Columns.Add("Salary");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
dataTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
dataTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//Assign data source
grid.DataSource = dataTable;
//Create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Assign string format to a row
for (int i = 0; i < grid.Columns.Count; i++)
{
grid.Rows[2].Cells[i].StringFormat = stringFormat;
}

```

```

}
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Save the document and close the instance of PdfDocument
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add page to the document
Dim page As PdfPage = document.Pages.Add
'Create a new PdfGrid
Dim grid As PdfGrid = New PdfGrid
'Create a DataTable
Dim dataTable As DataTable = New DataTable
'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
dataTable.Columns.Add("Salary")
'Add rows to the DataTable
dataTable.Rows.Add(New Object() {"E01", "Clay", "$10,000"})
dataTable.Rows.Add(New Object() {"E02", "Thomas", "$10,500"})
dataTable.Rows.Add(New Object() {"E03", "Simon", "$12,000"})
'Assign data source
grid.DataSource = dataTable
'Create and customize the string formats
Dim stringFormat As PdfStringFormat = New PdfStringFormat
stringFormat.Alignment = PdfTextAlignment.Center
stringFormat.LineAlignment = PdfVerticalAlignment.Middle
stringFormat.CharacterSpacing = 2.0F
'Assign string format to a row
Dim i As Integer = 0
For i = 0 To grid.Columns.Count - 1 Step 1
grid.Rows(2).Cells(i).StringFormat = stringFormat
Next
'Draw the PdfGrid on page
grid.Draw(page, New PointF(10, 10))
'Save the document and close the instance of PdfDocument
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the PdfDocument
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };

```

```

data.Add(grid1row1);
data.Add(grid1row2);
data.Add(grid1row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
grid.DataSource = dataTable;
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Assign string format to a row
for (int i = 0; i < grid.Columns.Count; i++)
{
    grid.Rows[2].Cells[i].StringFormat = stringFormat;
}
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Create memory stream
MemoryStream ms = new MemoryStream();
//Open the document in browser after saving it
document.Save(ms);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respective code samples
Save(ms, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the PdfDocument
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object grid1row1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object grid1row2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object grid1row3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(grid1row1);
data.Add(grid1row2);
data.Add(grid1row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
grid.DataSource = dataTable;
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Assign string format to a row
for (int i = 0; i < grid.Columns.Count; i++)

```

```

{
    grid.Rows[2].Cells[i].StringFormat = stringFormat;
}
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileName = "Output.pdf";
return fileStreamResult;

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add page to the PdfDocument
PdfPage page = document.Pages.Add();
//Create a new PdfGrid
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridlrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridlrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridlrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridlrow1);
data.Add(gridlrow2);
data.Add(gridlrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source
grid.DataSource = dataTable;
//create and customize the string formats
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Assign string format to a row
for (int i = 0; i < grid.Columns.Count; i++)
{
    grid.Rows[2].Cells[i].StringFormat = stringFormat;
}
//Draw the PdfGrid on page
grid.Draw(page, new PointF(10, 10));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file

```

```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Row and Column spanning

Essential PDF supports both row spanning and column spanning in a PDF table.

Row spanning in PdfGrid

You can span a row in [PdfGrid](#) by using the [RowSpan](#) property of [PdfGridCell](#). The following code snippet illustrates this.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create the page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Middle;
//Add columns to PdfGrid
pdfGrid.Columns.Add(5);
//Add rows to PdfGrid
for (int i = 0; i < pdfGrid.Columns.Count; i++)
{
PdfGridRow row = pdfGrid.Rows.Add();
row.Height = 20;
}
//Add RowSpan
PdfGridCell gridCell = pdfGrid.Rows[1].Cells[3];
gridCell.RowSpan = 3;
gridCell.StringFormat = format;
gridCell.Value = "Row Span";
gridCell.Style.BackgroundBrush = PdfBrushes.Yellow;
//Draw the PdfGrid
pdfGrid.Draw(page, new PointF(10, 10));
//Save the document
document.Save("Output.pdf");
//Close the document
document.Close(true);
```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Create the page
Dim page As PdfPage = document.Pages.Add
'Create a PdfGrid
Dim pdfGrid As PdfGrid = New PdfGrid
'Create and customize the string formats
Dim format As PdfStringFormat = New PdfStringFormat
format.Alignment = PdfTextAlignment.Center
format.LineAlignment = PdfVerticalAlignment.Middle
'Add columns to PdfGrid
pdfGrid.Columns.Add(5)
'Add rows to PdfGrid
For i = 0 To pdfGrid.Columns.Count - 1 Step 1
Dim row As PdfGridRow = pdfGrid.Rows.Add
row.Height = 20
Next
'Add RowSpan
Dim gridCell As PdfGridCell = pdfGrid.Rows(1).Cells(3)
gridCell.RowSpan = 3
gridCell.StringFormat = format
gridCell.Value = "Row Span"
gridCell.Style.BackgroundBrush = PdfBrushes.Yellow
'Draw the PdfGrid
pdfGrid.Draw(page, New PointF(10, 10))
'Save the document
document.Save("Output.pdf")
'Close the document
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create the page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Middle;
//Add columns to PdfGrid
pdfGrid.Columns.Add(5);
//Add rows to PdfGrid
for (int i = 0; i < pdfGrid.Columns.Count; i++)
{
PdfGridRow row = pdfGrid.Rows.Add();
row.Height = 20;
}
//Add RowSpan
PdfGridCell gridCell = pdfGrid.Rows[1].Cells[3];
gridCell.RowSpan = 3;
gridCell.StringFormat = format;

```

```

gridCell.Value = "Row Span";
gridCell.Style.BackgroundBrush = PdfBrushes.Yellow;
//Draw the PdfGrid
pdfGrid.Draw(page, new PointF(10, 10));
//Create memory stream
MemoryStream ms = new MemoryStream();
//Open the document in browser after saving it
document.Save(ms);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(ms, "Sample.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create the page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Middle;
//Add columns to PdfGrid
pdfGrid.Columns.Add(5);
//Add rows to PdfGrid
for (int i = 0; i < pdfGrid.Columns.Count; i++)
{
    PdfGridRow row = pdfGrid.Rows.Add();
    row.Height = 20;
}
//Add RowSpan
PdfGridCell gridCell = pdfGrid.Rows[1].Cells[3];
gridCell.RowSpan = 3;
gridCell.StringFormat = format;
gridCell.Value = "Row Span";
gridCell.Style.BackgroundBrush = PdfBrushes.Yellow;
//Draw the PdfGrid
pdfGrid.Draw(page, new PointF(10, 10));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Sample.pdf";
return fileStreamResult;

```

XAMARIN

```

//Create a new PDF document

```

```

PdfDocument document = new PdfDocument();
//Create the page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Middle;
//Add columns to PdfGrid
pdfGrid.Columns.Add(5);
//Add rows to PdfGrid
for (int i = 0; i < pdfGrid.Columns.Count; i++)
{
    PdfGridRow row = pdfGrid.Rows.Add();
    row.Height = 20;
}
//Add RowSpan
PdfGridCell gridCell = pdfGrid.Rows[1].Cells[3];
gridCell.RowSpan = 3;
gridCell.StringFormat = format;
gridCell.Value = "Row Span";
gridCell.Style.BackgroundBrush = PdfBrushes.Yellow;
//Draw the PdfGrid
pdfGrid.Draw(page, new PointF(10, 10));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Column spanning in PdfGrid

You can span a column in [PdfGrid](#) by using the [ColumnSpan](#) property of [PdfGridCell](#). The following code snippet explains this.

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create the page
PdfPage page = document.Pages.Add();

```



```

//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Middle;
//Add columns to PdfGrid
pdfGrid.Columns.Add(5);
//Add rows to PdfGrid
for (int i = 0; i < pdfGrid.Columns.Count; i++)
{
    PdfGridRow row = pdfGrid.Rows.Add();
    row.Height = 20;
}
//Add ColumnSpan
PdfGridCell gridCell = pdfGrid.Rows[1].Cells[0];
gridCell.ColumnSpan = 2;
gridCell.StringFormat = format;
gridCell.Value = "Column Span";
gridCell.Style.BackgroundBrush = PdfBrushes.Yellow;
//Draw the PdfGrid
pdfGrid.Draw(page, new PointF(10, 10));
//Save the document
document.Save("Output.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Create the page
Dim page As PdfPage = document.Pages.Add
'Create a PdfGrid
Dim pdfGrid As PdfGrid = New PdfGrid
'Create and customize the string formats
Dim format As PdfStringFormat = New PdfStringFormat
format.Alignment = PdfTextAlignment.Center
format.LineAlignment = PdfVerticalAlignment.Middle
'Add columns to PdfGrid
pdfGrid.Columns.Add(5)
'Add rows to PdfGrid
For i = 0 To pdfGrid.Columns.Count - 1 Step 1
    Dim row As PdfGridRow = pdfGrid.Rows.Add
    row.Height = 20
Next
'Add ColumnSpan
Dim gridCell As PdfGridCell = pdfGrid.Rows(1).Cells(0)
gridCell.ColumnSpan = 2
gridCell.StringFormat = format
gridCell.Value = "Column Span"
gridCell.Style.BackgroundBrush = PdfBrushes.Yellow
'Draw the PdfGrid
pdfGrid.Draw(page, New PointF(10, 10))
'Save the document
document.Save("Output.pdf")

```

```
'Close the document
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create the page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Middle;
//Add columns to PdfGrid
pdfGrid.Columns.Add(5);
//Add rows to PdfGrid
for (int i = 0; i < pdfGrid.Columns.Count; i++)
{
    PdfGridRow row = pdfGrid.Rows.Add();
    row.Height = 20;
}
//Add ColumnSpan
PdfGridCell gridCell = pdfGrid.Rows[1].Cells[0];
gridCell.ColumnSpan = 2;
gridCell.StringFormat = format;
gridCell.Value = "Column Span";
gridCell.Style.BackgroundBrush = PdfBrushes.Yellow;
//Draw the PdfGrid
pdfGrid.Draw(page, new PointF(10, 10));
//Create memory stream
MemoryStream ms = new MemoryStream();
//Open the document in browser after saving it
document.Save(ms);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(ms, "Sample.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create the page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Middle;
//Add columns to PdfGrid
pdfGrid.Columns.Add(5);
//Add rows to PdfGrid
```

```

for (int i = 0; i < pdfGrid.Columns.Count; i++)
{
    PdfGridRow row = pdfGrid.Rows.Add();
    row.Height = 20;
}
//Add ColumnSpan
PdfGridCell gridCell = pdfGrid.Rows[1].Cells[0];
gridCell.ColumnSpan = 2;
gridCell.StringFormat = format;
gridCell.Value = "Column Span";
gridCell.Style.BackgroundBrush = PdfBrushes.Yellow;
//Draw the PdfGrid
pdfGrid.Draw(page, new PointF(10, 10));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Sample.pdf";
return fileStreamResult;

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create the page
PdfPage page = document.Pages.Add();
//Create a PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Create and customize the string formats
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Middle;
//Add columns to PdfGrid
pdfGrid.Columns.Add(5);
//Add rows to PdfGrid
for (int i = 0; i < pdfGrid.Columns.Count; i++)
{
    PdfGridRow row = pdfGrid.Rows.Add();
    row.Height = 20;
}
//Add ColumnSpan
PdfGridCell gridCell = pdfGrid.Rows[1].Cells[0];
gridCell.ColumnSpan = 2;
gridCell.StringFormat = format;
gridCell.Value = "Column Span";
gridCell.Style.BackgroundBrush = PdfBrushes.Yellow;
//Draw the PdfGrid
pdfGrid.Draw(page, new PointF(10, 10));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document

```

```
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Table cell styles

Essential PDF allows you to add different styles like background color using [BackgroundBrush](#), background image using [BackgroundImage](#), border using [Borders](#), cell dimension by setting row [Height](#) and column [Width](#), along with spanning through [RowSpan](#) and [ColumnSpan](#).

The following code snippet explains how to add different cell styles to a cell in [PdfGrid](#).

C#

```
//Create a new PDF document
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid instance
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
dataTable.Columns.Add("Salary");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
dataTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
dataTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//Assign data source to PdfGrid
pdfGrid.DataSource = dataTable;
//Assign row height and column width
pdfGrid.Rows[1].Height = 50;
pdfGrid.Columns[1].Width = 100;
//Initialize PdfGridCellStyle and set background color
PdfGridCellStyle gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundBrush = PdfBrushes.Yellow;
//Assign background color to a PdfGridCell
PdfGridCell gridCell = pdfGrid.Rows[0].Cells[0];
gridCell.Style = gridCellStyle;
//Initialize PdfGridCellStyle and set background image
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundImage = PdfImage.FromFile("Autumn Leaves.jpg");
//Assign background image to a PdfGridCell
```

```

gridCell = pdfGrid.Rows[1].Cells[1];
gridCell.Style = gridCellStyle;
gridCell.ImagePosition = PdfGridImagePosition.Fit;
//Initialize PdfGridCellStyle and set the border color
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.Borders.All = PdfPens.Red;
//Assign the border color to a PdfGridCell
gridCell = pdfGrid.Rows[2].Cells[2];
gridCell.Style = gridCellStyle;
//Initialize PdfGridCellStyle and set text color
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.TextBrush = PdfBrushes.DarkBlue;
//Assign text color to a PdfGridCell
gridCell = pdfGrid.Rows[0].Cells[2];
gridCell.Style = gridCellStyle;
//Set the column span
pdfGrid.Rows[2].Cells[0].ColumnSpan = 2;
//Initialize PdfStringFormat and set the properties
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
//Set the PdfStringFormat to PdfGridCell
gridCell = pdfGrid.Rows[2].Cells[0];
gridCell.StringFormat = stringFormat;
//Draw the table in the PDF page
pdfGrid.Draw(pdfPage, new PointF(10, 10));
//Save the document
pdfDocument.Save("Output.pdf");
//Close the instance of PdfDocument
pdfDocument.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim pdfDocument As PdfDocument = New PdfDocument
Dim pdfPage As PdfPage = pdfDocument.Pages.Add
'Create a new PdfGrid instance
Dim pdfGrid As PdfGrid = New PdfGrid
'Create a DataTable
Dim dataTable As DataTable = New DataTable
'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
dataTable.Columns.Add("Salary")
'Add rows to the DataTable
dataTable.Rows.Add(New Object() {"E01", "Clay", "$10,000"})
dataTable.Rows.Add(New Object() {"E02", "Thomas", "$10,500"})
dataTable.Rows.Add(New Object() {"E03", "Simon", "$12,000"})
'Assign data source to PdfGrid
pdfGrid.DataSource = dataTable
'Assign row height and column width
pdfGrid.Rows(1).Height = 50
pdfGrid.Columns(1).Width = 100
'Initialize PdfGridCellStyle and set background color
Dim gridCellStyle As PdfGridCellStyle = New PdfGridCellStyle
gridCellStyle.BackgroundBrush = PdfBrushes.Yellow

```

```

'Assign background color to a PdfGridCell
Dim gridCell As PdfGridCell = pdfGrid.Rows(0).Cells(0)
gridCell.Style = gridCellStyle
'Initialize PdfGridCellStyle and set background image
gridCellStyle = New PdfGridCellStyle
gridCellStyle.BackgroundImage = PdfImage.FromFile("Autumn Leaves.jpg")
'Assign background image to a PdfGridCell
gridCell = pdfGrid.Rows(1).Cells(1)
gridCell.Style = gridCellStyle
gridCell.ImagePosition = PdfGridImagePosition.Fit
'Initialize PdfGridCellStyle and set the border color
gridCellStyle = New PdfGridCellStyle
gridCellStyle.Borders.All = PdfPens.Red
'Assign the border color to a PdfGridCell
gridCell = pdfGrid.Rows(2).Cells(2)
gridCell.Style = gridCellStyle
'Initialize PdfGridCellStyle and set text color
gridCellStyle = New PdfGridCellStyle
gridCellStyle.TextBrush = PdfBrushes.DarkBlue
'Assign text color to a PdfGridCell
gridCell = pdfGrid.Rows(0).Cells(2)
gridCell.Style = gridCellStyle
'Set the column span
pdfGrid.Rows(2).Cells(0).ColumnSpan = 2
'Initialize PdfStringFormat and set the properties
Dim stringFormat As PdfStringFormat = New PdfStringFormat
stringFormat.Alignment = PdfTextAlignment.Center
stringFormat.LineAlignment = PdfVerticalAlignment.Middle
'Set the PdfStringFormat to PdfGridCell
gridCell = pdfGrid.Rows(2).Cells(0)
gridCell.StringFormat = stringFormat
'Draw the table in the PDF page
pdfGrid.Draw(pdfPage, New PointF(10, 10))
'Save the document
pdfDocument.Save("Output.pdf")
'Close the instance of PdfDocument
pdfDocument.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid instance
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridlrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridlrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridlrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridlrow1);
data.Add(gridlrow2);
data.Add(gridlrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source to PdfGrid

```

```

pdfGrid.DataSource = dataTable;
//Assign row height and column width
pdfGrid.Rows[1].Height = 50;
pdfGrid.Columns[1].Width = 100;
//Initialize PdfGridCellStyle and set background color
PdfGridCellStyle gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundBrush = PdfBrushes.Yellow;
//Assign background color to a PdfGridCell
PdfGridCell gridCell = pdfGrid.Rows[0].Cells[0];
gridCell.Style = gridCellStyle;
//Initialize PdfGridCellStyle and set background image
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Da
ta.Autumn Leaves.jpg");
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundImage = PdfImage.FromStream(imageStream);
//Assign background image to a PdfGridCell
gridCell = pdfGrid.Rows[1].Cells[1];
gridCell.Style = gridCellStyle;
gridCell.ImagePosition = PdfGridImagePosition.Fit;
//Initialize PdfGridCellStyle and set the border color
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.Borders.All = PdfPens.Red;
//Assign the border color to a PdfGridCell
gridCell = pdfGrid.Rows[2].Cells[2];
gridCell.Style = gridCellStyle;
//Initialize PdfGridCellStyle and set text color
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.TextBrush = PdfBrushes.DarkBlue;
//Assign text color to a PdfGridCell
gridCell = pdfGrid.Rows[0].Cells[2];
gridCell.Style = gridCellStyle;
//Set the column span
pdfGrid.Rows[2].Cells[0].ColumnSpan = 2;
//Initialize PdfStringFormat and set the properties
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
//Set the PdfStringFormat to PdfGridCell
gridCell = pdfGrid.Rows[2].Cells[0];
gridCell.StringFormat = stringFormat;
//Draw the table in the PDF page
pdfGrid.Draw(pdfPage, new PointF(10, 10));
//Create memory stream
MemoryStream ms = new MemoryStream();
//Open the document in browser after saving it
pdfDocument.Save(ms);
//Close the document
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(ms, "Output.pdf");

```

ASP.NET CORE

```
//Create a new PDF document
```

```

PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid instance
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridlrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridlrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridlrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridlrow1);
data.Add(gridlrow2);
data.Add(gridlrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source to PdfGrid
pdfGrid.DataSource = dataTable;
//Assign row height and column width
pdfGrid.Rows[1].Height = 50;
pdfGrid.Columns[1].Width = 100;
//Initialize PdfGridCellStyle and set background color
PdfGridCellStyle gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundBrush = PdfBrushes.Yellow;
//Assign background color to a PdfGridCell
PdfGridCell gridCell = pdfGrid.Rows[0].Cells[0];
gridCell.Style = gridCellStyle;
//Initialize PdfGridCellStyle and set background image
FileStream imageStream = new FileStream("Autumn Leaves.jpg", FileMode.Open);
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundImage = PdfImage.FromStream(imageStream);
//Assign background image to a PdfGridCell
gridCell = pdfGrid.Rows[1].Cells[1];
gridCell.Style = gridCellStyle;
gridCell.ImagePosition = PdfGridImagePosition.Fit;
//Initialize PdfGridCellStyle and set the border color
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.Borders.All = PdfPens.Red;
//Assign the border color to a PdfGridCell
gridCell = pdfGrid.Rows[2].Cells[2];
gridCell.Style = gridCellStyle;
//Initialize PdfGridCellStyle and set text color
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.TextBrush = PdfBrushes.DarkBlue;
//Assign text color to a PdfGridCell
gridCell = pdfGrid.Rows[0].Cells[2];
gridCell.Style = gridCellStyle;
//Set the column span
pdfGrid.Rows[2].Cells[0].ColumnSpan = 2;
//Initialize PdfStringFormat and set the properties
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
//Set the PdfStringFormat to PdfGridCell
gridCell = pdfGrid.Rows[2].Cells[0];
gridCell.StringFormat = stringFormat;
//Draw the table in the PDF page
pdfGrid.Draw(pdfPage, new PointF(10, 10));
//Saving the PDF to the MemoryStream

```



```

MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;

```

XAMARIN

```

//Create a new PDF document
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid instance
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object grid1row1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object grid1row2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object grid1row3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(grid1row1);
data.Add(grid1row2);
data.Add(grid1row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source to PdfGrid
pdfGrid.DataSource = dataTable;
//Assign row height and column width
pdfGrid.Rows[1].Height = 50;
pdfGrid.Columns[1].Width = 100;
//Initialize PdfGridCellStyle and set background color
PdfGridCellStyle gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundBrush = PdfBrushes.Yellow;
//Assign background color to a PdfGridCell
PdfGridCell gridCell = pdfGrid.Rows[0].Cells[0];
gridCell.Style = gridCellStyle;
//Initialize PdfGridCellStyle and set background image
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Data.Autumn Leaves.jpg");
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundImage = PdfImage.FromStream(imageStream);
//Assign background image to a PdfGridCell
gridCell = pdfGrid.Rows[1].Cells[1];
gridCell.Style = gridCellStyle;
gridCell.ImagePosition = PdfGridImagePosition.Fit;
//Initialize PdfGridCellStyle and set the border color
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.Borders.All = PdfPens.Red;
//Assign the border color to a PdfGridCell
gridCell = pdfGrid.Rows[2].Cells[2];
gridCell.Style = gridCellStyle;
//Initialize PdfGridCellStyle and set text color
gridCellStyle = new PdfGridCellStyle();

```

```

gridCellStyle.TextBrush = PdfBrushes.DarkBlue;
//Assign text color to a PdfGridCell
gridCell = pdfGrid.Rows[0].Cells[2];
gridCell.Style = gridCellStyle;
//Set the column span
pdfGrid.Rows[2].Cells[0].ColumnSpan = 2;
//Initialize PdfStringFormat and set the properties
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
//Set the PdfStringFormat to PdfGridCell
gridCell = pdfGrid.Rows[2].Cells[0];
gridCell.StringFormat = stringFormat;
//Draw the table in the PDF page
pdfGrid.Draw(pdfPage, new PointF(10, 10));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Close the document
pdfDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Table row style

Essential PDF supports applying different styles to a row in [PdfGrid](#) by using [PdfGridCellStyle](#) and [PdfGridRow](#) instances. The following code snippet explains this.

C#

```

//Create a new PDF document
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid instance
PdfGrid pdfGrid = new PdfGrid();
//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
dataTable.Columns.Add("Salary");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
dataTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });

```

```

dataTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//Assign data source to PdfGrid
pdfGrid.DataSource = dataTable;
//Assign row height and column width
pdfGrid.Rows[1].Height = 50;
pdfGrid.Columns[1].Width = 100;
//Initialize PdfStringFormat and set the properties
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Initialize PdfGridCellStyle. Set background color and string format
PdfGridCellStyle gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundBrush = PdfBrushes.Yellow;
gridCellStyle.StringFormat = stringFormat;
//Initialize PdfGridRow and apply PdfGridCellStyle to the row
PdfGridRow gridRow = pdfGrid.Rows[0];
gridRow.ApplyStyle(gridCellStyle);
//Initialize PdfGridCellStyle and set background image
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundImage = PdfImage.FromFile("Autumn Leaves.jpg");
//Initialize PdfGridRow and apply PdfGridCellStyle to the row
gridRow = pdfGrid.Rows[1];
gridRow.ApplyStyle(gridCellStyle);
//Initialize PdfGridCellStyle. Set the border color and text color
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.Borders.All = PdfPens.Red;
gridCellStyle.TextBrush = PdfBrushes.DarkBlue;
//Initialize PdfGridRow and apply PdfGridCellStyle to the row
gridRow = pdfGrid.Rows[2];
gridRow.ApplyStyle(gridCellStyle);
//Draw the table in the PDF page
pdfGrid.Draw(pdfPage, new PointF(10, 10));
//Save the document
pdfDocument.Save("Output.pdf");
//Close the instance of PdfDocument
pdfDocument.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim pdfDocument As PdfDocument = New PdfDocument
Dim pdfPage As PdfPage = pdfDocument.Pages.Add
'Create a new PdfGrid instance
Dim pdfGrid As PdfGrid = New PdfGrid
'Create a DataTable
Dim dataTable As DataTable = New DataTable
'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
dataTable.Columns.Add("Salary")
'Add rows to the DataTable
dataTable.Rows.Add(New Object() {"E01", "Clay", "$10,000"})
dataTable.Rows.Add(New Object() {"E02", "Thomas", "$10,500"})
dataTable.Rows.Add(New Object() {"E03", "Simon", "$12,000"})
'Assign data source to PdfGrid

```

```

pdfGrid.DataSource = dataTable
'Assign row height and column width
pdfGrid.Rows(1).Height = 50
pdfGrid.Columns(1).Width = 100
'Initialize PdfStringFormat and set the properties
Dim stringFormat As PdfStringFormat = New PdfStringFormat
stringFormat.Alignment = PdfTextAlignment.Center
stringFormat.LineAlignment = PdfVerticalAlignment.Middle
stringFormat.CharacterSpacing = 2.0F
'Initialize PdfGridCellStyle. Set background color and string format
Dim gridCellStyle As PdfGridCellStyle = New PdfGridCellStyle
gridCellStyle.BackgroundBrush = PdfBrushes.Yellow
gridCellStyle.StringFormat = stringFormat
'Initialize PdfGridRow and apply PdfGridCellStyle to the row
Dim gridRow As PdfGridRow = pdfGrid.Rows(0)
gridRow.ApplyStyle(gridCellStyle)
'Initialize PdfGridCellStyle and set background image
gridCellStyle = New PdfGridCellStyle
gridCellStyle.BackgroundImage = PdfImage.FromFile("Autumn Leaves.jpg")
'Initialize PdfGridRow and apply PdfGridCellStyle to the row
gridRow = pdfGrid.Rows(1)
gridRow.ApplyStyle(gridCellStyle)
'Initialize PdfGridCellStyle. Set the border color and text color
gridCellStyle = New PdfGridCellStyle
gridCellStyle.Borders.All = PdfPens.Red
gridCellStyle.TextBrush = PdfBrushes.DarkBlue
'Initialize PdfGridRow and apply PdfGridCellStyle to the row
gridRow = pdfGrid.Rows(2)
gridRow.ApplyStyle(gridCellStyle)
'Draw the table in the PDF page
pdfGrid.Draw(pdfPage, New PointF(10, 10))
'Save the document
pdfDocument.Save("Output.pdf")
'Close the instance of PdfDocument
pdfDocument.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid instance
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridlrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridlrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridlrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridlrow1);
data.Add(gridlrow2);
data.Add(gridlrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source to PdfGrid
pdfGrid.DataSource = dataTable;
//Assign row height and column width

```

```

pdfGrid.Rows[1].Height = 50;
pdfGrid.Columns[1].Width = 100;
//Initialize PdfStringFormat and set the properties
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Initialize PdfGridCellStyle. Set background color and string format
PdfGridCellStyle gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundBrush = PdfBrushes.Yellow;
gridCellStyle.StringFormat = stringFormat;
//Initialize PdfGridRow and apply PdfGridCellStyle to the row
PdfGridRow gridRow = pdfGrid.Rows[0];
gridRow.ApplyStyle(gridCellStyle);
//Initialize PdfGridCellStyle and set background image
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Data.Autumn Leaves.jpg");
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundImage = PdfImage.FromStream(imageStream);
//Initialize PdfGridRow and apply PdfGridCellStyle to the row
gridRow = pdfGrid.Rows[1];
gridRow.ApplyStyle(gridCellStyle);
//Initialize PdfGridCellStyle. Set the border color and text color
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.Borders.All = PdfPens.Red;
gridCellStyle.TextBrush = PdfBrushes.DarkBlue;
//Initialize PdfGridRow and apply PdfGridCellStyle to the row
gridRow = pdfGrid.Rows[2];
gridRow.ApplyStyle(gridCellStyle);
//Draw the table in the PDF page
pdfGrid.Draw(pdfPage, new PointF(10, 10));
//Create memory stream
MemoryStream ms = new MemoryStream();
//Open the document in browser after saving it
pdfDocument.Save(ms);
//Close the document
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(ms, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid instance
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridlrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridlrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridlrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridlrow1);
data.Add(gridlrow2);

```

```

data.Add(grid1row3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source to PdfGrid
pdfGrid.DataSource = dataTable;
//Assign row height and column width
pdfGrid.Rows[1].Height = 50;
pdfGrid.Columns[1].Width = 100;
//Initialize PdfStringFormat and set the properties
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Initialize PdfGridCellStyle. Set background color and string format
PdfGridCellStyle gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundBrush = PdfBrushes.Yellow;
gridCellStyle.StringFormat = stringFormat;
//Initialize PdfGridRow and apply PdfGridCellStyle to the row
PdfGridRow gridRow = pdfGrid.Rows[0];
gridRow.ApplyStyle(gridCellStyle);
//Initialize PdfGridCellStyle and set background image
FileStream imageStream = new FileStream("Autumn Leaves.jpg", FileMode.Open);
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundImage = PdfImage.FromStream(imageStream);
//Initialize PdfGridRow and apply PdfGridCellStyle to the row
gridRow = pdfGrid.Rows[1];
gridRow.ApplyStyle(gridCellStyle);
//Initialize PdfGridCellStyle. Set the border color and text color
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.Borders.All = PdfPens.Red;
gridCellStyle.TextBrush = PdfBrushes.DarkBlue;
//Initialize PdfGridRow and apply PdfGridCellStyle to the row
gridRow = pdfGrid.Rows[2];
gridRow.ApplyStyle(gridCellStyle);
//Draw the table in the PDF page
pdfGrid.Draw(pdfPage, new PointF(10, 10));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;

```

XAMARIN

```

//Create a new PDF document
PdfDocument pdfDocument = new PdfDocument();
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a new PdfGrid instance
PdfGrid pdfGrid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();

```

```

Object gridrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridrow1);
data.Add(gridrow2);
data.Add(gridrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign data source to PdfGrid
pdfGrid.DataSource = dataTable;
//Assign row height and column width
pdfGrid.Rows[1].Height = 50;
pdfGrid.Columns[1].Width = 100;
//Initialize PdfStringFormat and set the properties
PdfStringFormat stringFormat = new PdfStringFormat();
stringFormat.Alignment = PdfTextAlignment.Center;
stringFormat.LineAlignment = PdfVerticalAlignment.Middle;
stringFormat.CharacterSpacing = 2f;
//Initialize PdfGridCellStyle. Set background color and string format
PdfGridCellStyle gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundBrush = PdfBrushes.Yellow;
gridCellStyle.StringFormat = stringFormat;
//Initialize PdfGridRow and apply PdfGridCellStyle to the row
PdfGridRow gridRow = pdfGrid.Rows[0];
gridRow.ApplyStyle(gridCellStyle);
//Initialize PdfGridCellStyle and set background image
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Data.Autumn Leaves.jpg");
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.BackgroundImage = PdfImage.FromStream(imageStream);
//Initialize PdfGridRow and apply PdfGridCellStyle to the row
gridRow = pdfGrid.Rows[1];
gridRow.ApplyStyle(gridCellStyle);
//Initialize PdfGridCellStyle. Set the border color and text color
gridCellStyle = new PdfGridCellStyle();
gridCellStyle.Borders.All = PdfPens.Red;
gridCellStyle.TextBrush = PdfBrushes.DarkBlue;
//Initialize PdfGridRow and apply PdfGridCellStyle to the row
gridRow = pdfGrid.Rows[2];
gridRow.ApplyStyle(gridCellStyle);
//Draw the table in the PDF page
pdfGrid.Draw(pdfPage, new PointF(10, 10));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Close the document
pdfDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}

```

```

}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
    "application/pdf", stream);
}

```

Difference between PdfLightTable and PdfGrid

Both the [PdfGrid](#) and [PdfLightTable](#) models are supported across all the platforms and the below table explains the level of customizations both the models provide.

Features	PdfLightTable	PdfGrid
Formatting		
Row	No direct API, possible through events.	Yes
Column	Yes (StringFormat)	Yes (StringFormat)
Cell	No direct API for single cell formatting, possible through events.	Yes
Others		
Row span	No	Yes
Column span	No direct API, possible through events.	Yes
Nested Grid	Possible through events	Yes
Layout Events	BeginCellLayout, BeginPageLayout, BeginRowLayout, EndCellLayout, EndPageLayout, EndRowLayout	BeginPageLayout, EndPageLayout

Working with flow layout

PDF documents can be created using flow model instead of adding elements through absolute positioning. To create a PDF document in flow model, please add references to the following set of assemblies.

Assembly Name	Description
Syncfusion.Pdf.Base	This assembly contains the core feature for creating, manipulating and saving PDF documents.
Syncfusion.Compression.Base	This assembly is required for compressing the internal contents of a PDF document.

Syncfusion.DocIO.Base	This assembly contains the core features needed for creating, reading, manipulating a Word document.
Syncfusion.DocToPdfConverter.Base	This assembly is required for converting word to PDF

Include the following namespaces in your .cs or .vb file as shown below.

C#

```
using Syncfusion.Pdf;
using Syncfusion.DocIO.DLS;
using Syncfusion.DocToPDFConverter;
```

VB.NET

```
Imports Syncfusion.Pdf
Imports Syncfusion.DocIO.DLS
Imports Syncfusion.DocToPDFConverter
```

ASP.NET CORE

```
using Syncfusion.Pdf;
using Syncfusion.DocIO.DLS;
using Syncfusion.DocIORenderer;
```

XAMARIN

```
using Syncfusion.Pdf;
using Syncfusion.DocIO.DLS;
using Syncfusion.DocIORenderer;
```

Working with Text

You can create a PDF document with multiple paragraph text using flow model with the following code snippet.

C#

```
//Creates a new Word document
WordDocument wordDocument = new WordDocument();
//Adds new section with single paragraph to the document
wordDocument.EnsureMinimal();
wordDocument.LastSection.PageSetup.Margins.All = 15;
//Get Last paragraph of the document
IWParagraph paragraph = wordDocument.LastParagraph;
//Create a custom style
WParagraphStyle paragraphStyle = wordDocument.Styles.FindByName("Normal") as
WParagraphStyle;
paragraphStyle.CharacterFormat.Font = new Font("Times New Roman", 12);
paragraphStyle.ParagraphFormat.HorizontalAlignment =
HorizontalAlignment.Justify;
paragraphStyle.ParagraphFormat.AfterSpacing = 15f;
WSection section = wordDocument.LastSection;
```

```

string text = "Adventure Works Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Adds new text to the paragraph
paragraph.AppendText(text);
//Adds first text to the paragraph
paragraph = section.AddParagraph();
paragraph.AppendText(text);
//Second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText(text);
//Creates an instance of the DocToPDFConverter
DocToPDFConverter converter = new DocToPDFConverter();
//Converts Word document into PDF document
PdfDocument pdfDocument = converter.ConvertToPDF(wordDocument);
//Save and close the PDF document
pdfDocument.Save("Output.pdf");
pdfDocument.Close(true);
//Close the document
wordDocument.Close();

```

VB.NET

```

'Creates a new Word document
Dim wordDocument As New WordDocument()
'Adds new section with single paragraph to the document
wordDocument.EnsureMinimal()
wordDocument.LastSection.PageSetup.Margins.All = 15
'Get Last paragraph of the document
Dim paragraph As IParagraph = wordDocument.LastParagraph
'Create a custom style
Dim paragraphStyle As WParagraphStyle =
TryCast(wordDocument.Styles.FindByName("Normal"), WParagraphStyle)
paragraphStyle.CharacterFormat.Font = New Font("Times New Roman", 12)
paragraphStyle.ParagraphFormat.HorizontalAlignment =
HorizontalAlignment.Justify
paragraphStyle.ParagraphFormat.AfterSpacing = 15F
Dim section As WSection = wordDocument.LastSection
Dim text As String = "Adventure Works Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company. The company manufactures and sells
metal and composite bicycles to North American, European and Asian
commercial markets. While its base operation is located in Washington with
290 employees, several regional sales teams are located throughout their
market base."
'Adds new text to the paragraph
paragraph.AppendText(text)
'Adds first text to the paragraph
paragraph = section.AddParagraph()
paragraph.AppendText(text)
'Second paragraph
paragraph = section.AddParagraph()
paragraph.AppendText(text)

```

```

'Creates an instance of the DocToPDFConverter
Dim converter As New DocToPDFConverter()
'Converts Word document into PDF document
Dim pdfDocument As PdfDocument = converter.ConvertToPDF(wordDocument)
'Save and close the PDF document
pdfDocument.Save("Output.pdf")
pdfDocument.Close(True)
'Close the document
wordDocument.Close()

```

ASP.NET CORE

```

//Creates a new Word document
WordDocument wordDocument = new WordDocument();
//Adds new section with single paragraph to the document
wordDocument.EnsureMinimal();
wordDocument.LastSection.PageSetup.Margins.All = 15;
//Get Last paragraph of the document
IWParagraph paragraph = wordDocument.LastParagraph;
//Create a custom style
WParagraphStyle paragraphStyle = wordDocument.Styles.FindByName("Normal") as
WParagraphStyle;
paragraphStyle.CharacterFormat.Font = new Font("Times New Roman", 12);
paragraphStyle.ParagraphFormat.HorizontalAlignment =
HorizontalAlignment.Justify;
paragraphStyle.ParagraphFormat.AfterSpacing = 15f;
WSection section = wordDocument.LastSection;
string text = "Adventure Works Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Adds new text to the paragraph
paragraph.AppendText(text);
//Adds first text to the paragraph
paragraph = section.AddParagraph();
paragraph.AppendText(text);
//Second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText(text);
//Instantiation of DocIORenderer for Word to PDF conversion
DocIORenderer render = new DocIORenderer();
//Converts Word document into PDF document
PdfDocument pdfDocument = render.ConvertToPDF(wordDocument);
//Releases all resources used by the Word document and DocIO Renderer
objects
render.Dispose();
wordDocument.Dispose();
//Save the document into stream.
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Close the documents.
pdfDocument.Close(true);
//Defining the ContentType for pdf file.

```

```

string contentType = "application/pdf";
//Define the file name.
string fileName = " Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

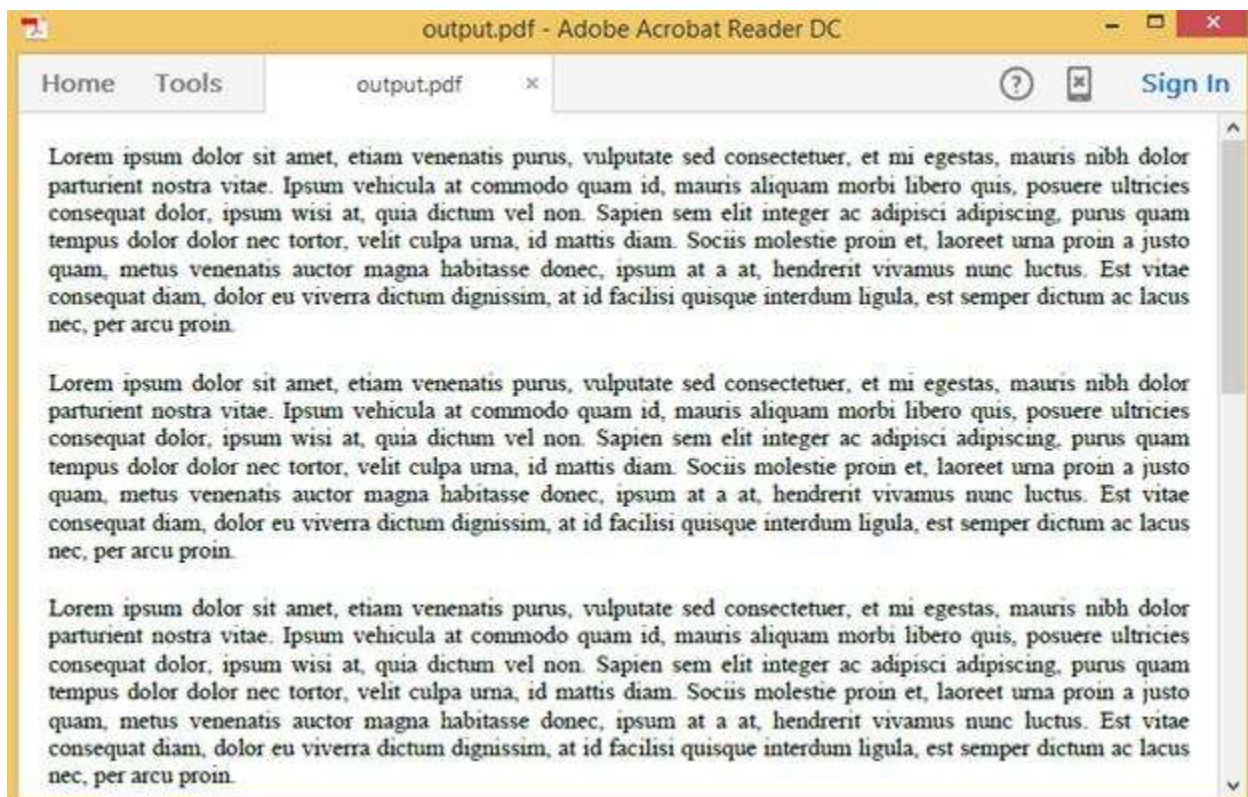
```

//Creates a new Word document
WordDocument wordDocument = new WordDocument();
//Adds new section with single paragraph to the document
wordDocument.EnsureMinimal();
wordDocument.LastSection.PageSetup.Margins.All = 15;
//Get Last paragraph of the document
IWParagraph paragraph = wordDocument.LastParagraph;
//Create a custom style
WParagraphStyle paragraphStyle = wordDocument.Styles.FindByName("Normal") as
WParagraphStyle;
paragraphStyle.CharacterFormat.Font = new Syncfusion.Drawing.Font("Times New
Roman", 12);
paragraphStyle.ParagraphFormat.HorizontalAlignment =
HorizontalAlignment.Justify;
paragraphStyle.ParagraphFormat.AfterSpacing = 15f;
WSection section = wordDocument.LastSection;
string text = "Adventure Works Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Adds new text to the paragraph
paragraph.AppendText(text);
//Adds first text to the paragraph
paragraph = section.AddParagraph();
paragraph.AppendText(text);
//Second paragraph
paragraph = section.AddParagraph();
paragraph.AppendText(text);
//Instantiation of DocIORenderer for Word to PDF conversion
DocIORenderer render = new DocIORenderer();
//Converts Word document into PDF document
PdfDocument pdfDocument = render.ConvertToPDF(wordDocument);
//Releases all resources used by the Word document and DocIO Renderer
objects
render.Dispose();
wordDocument.Dispose();
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Close the document
pdfDocument.Close();
//Save the stream into pdf file

```

```
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

The following image shows the PDF document with multiple paragraphs created using the flow model.



Working with text and images

You can create PDF document with text and image using the following code snippet.

C#

```
//A new document is created.
WordDocument document = new WordDocument();
//Adding a new section to the document.
WSection section = document.AddSection() as WSection;
//Set Margin of the section
section.PageSetup.Margins.All = 20;
//Create Paragraph styles
```

```

WParagraphStyle style = document.AddParagraphStyle("Normal") as
WParagraphStyle;
style.CharacterFormat.FontName = "Calibri";
style.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Justify;
style.CharacterFormat.FontSize = 11f;
style.ParagraphFormat.AfterSpacing = 8;
style.ParagraphFormat.FirstLineIndent = 36f;
style = document.AddParagraphStyle("Heading 1") as WParagraphStyle;
style.ApplyBaseStyle("Normal");
style.CharacterFormat.FontName = "Calibri Light";
style.CharacterFormat.FontSize = 16f;
style.CharacterFormat.TextColor = Color.FromArgb(46, 116, 181);
//Appends paragraph.
IWParagraph paragraph = section.AddParagraph();
paragraph.ApplyStyle("Heading 1");
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
paragraph.ParagraphFormat.AfterSpacing = 10;
WTextRange textRange = paragraph.AppendText("Adventure Works Cycles") as
WTextRange;
textRange.CharacterFormat.FontSize = 18f;
textRange.CharacterFormat.FontName = "Calibri";
string text =
"Adventure Works Cycles, the fictitious company on which the AdventureWorks
sample databases are based, is a large, multinational manufacturing company.
The company manufactures and sells metal and composite bicycles to North
American, European and Asian commercial markets. While its base operation is
located in Washington with 290 employees, several regional sales teams are
located throughout their market base.";
//Appends paragraph.
paragraph = section.AddParagraph();
textRange = paragraph.AppendText(text) as WTextRange;
paragraph = section.AddParagraph();
textRange = paragraph.AppendText(text) as WTextRange;
//Add image to the paragraph
paragraph = section.AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
paragraph.AppendPicture(Image.FromFile("Mountain-200.jpg"));
//Appends paragraph.
paragraph = section.AddParagraph();
textRange = paragraph.AppendText(text) as WTextRange;
//Creates an instance of the DocToPDFConverter
DocToPDFConverter converter = new DocToPDFConverter();
//Converts Word document into PDF document
PdfDocument pdfDocument = converter.ConvertToPDF(document);
//Saves the PDF file
pdfDocument.Save("Sample.pdf");
pdfDocument.Close(true);
document.Close();

```

VB.NET

```

'A new document is created.
Dim document As New WordDocument()

```

```

'Adding a new section to the document.
Dim section As WSection = TryCast(document.AddSection(), WSection)
'Set Margin of the section
section.PageSetup.Margins.All = 20
'Create Paragraph styles
Dim style As WParagraphStyle = TryCast(document.AddParagraphStyle("Normal"),
WParagraphStyle)
style.CharacterFormat.FontName = "Calibri"
style.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Justify
style.CharacterFormat.FontSize = 11F
style.ParagraphFormat.AfterSpacing = 8
style.ParagraphFormat.FirstLineIndent = 36F
style = TryCast(document.AddParagraphStyle("Heading 1"), WParagraphStyle)
style.ApplyBaseStyle("Normal")
style.CharacterFormat.FontName = "Calibri Light"
style.CharacterFormat.FontSize = 16F
style.CharacterFormat.TextColor = Color.FromArgb(46, 116, 181)
'Appends paragraph.
Dim paragraph As IWParagraph = section.AddParagraph()
paragraph.ApplyStyle("Heading 1")
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center
paragraph.ParagraphFormat.AfterSpacing = 10
Dim textRange As WTextRange = TryCast(paragraph.AppendText("Adventure Works
Cycles"), WTextRange)
textRange.CharacterFormat.FontSize = 18F
textRange.CharacterFormat.FontName = "Calibri"
Dim text As String = "Adventure Works Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company. The company manufactures and sells
metal and composite bicycles to North American, European and Asian
commercial markets. While its base operation is located in Washington with
290 employees, several regional sales teams are located throughout their
market base."
'Appends paragraph.
paragraph = section.AddParagraph()
textRange = TryCast(paragraph.AppendText(text), WTextRange)
paragraph = section.AddParagraph()
textRange = TryCast(paragraph.AppendText(text), WTextRange)
'Add image to the paragraph
paragraph = section.AddParagraph()
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center
paragraph.AppendPicture(Image.FromFile("Mountain-200.jpg"))
'Appends paragraph.
paragraph = section.AddParagraph()
textRange = TryCast(paragraph.AppendText(text), WTextRange)
'Creates an instance of the DocToPDFConverter
Dim converter As New DocToPDFConverter()
'Converts Word document into PDF document
Dim pdfDocument As PdfDocument = converter.ConvertToPDF(document)
'Saves the PDF file
pdfDocument.Save("Sample.pdf")
pdfDocument.Close(True)
document.Close()

```


ASP.NET CORE

```
//A new document is created.
WordDocument document = new WordDocument();
//Adding a new section to the document.
WSection section = document.AddSection() as WSection;
//Set Margin of the section
section.PageSetup.Margins.All = 20;
//Create Paragraph styles
WParagraphStyle style = document.AddParagraphStyle("Normal") as
WParagraphStyle;
style.CharacterFormat.FontName = "Calibri";
style.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Justify;
style.CharacterFormat.FontSize = 11f;
style.ParagraphFormat.AfterSpacing = 8;
style.ParagraphFormat.FirstLineIndent = 36f;
style = document.AddParagraphStyle("Heading 1") as WParagraphStyle;
style.ApplyBaseStyle("Normal");
style.CharacterFormat.FontName = "Calibri Light";
style.CharacterFormat.FontSize = 16f;
style.CharacterFormat.TextColor = Color.FromArgb(46, 116, 181);
//Appends paragraph.
IWParagraph paragraph = section.AddParagraph();
paragraph.ApplyStyle("Heading 1");
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
paragraph.ParagraphFormat.AfterSpacing = 10;
WTextRange textRange = paragraph.AppendText("Adventure Works Cycles") as
WTextRange;
textRange.CharacterFormat.FontSize = 18f;
textRange.CharacterFormat.FontName = "Calibri";
string text =
"Adventure Works Cycles, the fictitious company on which the AdventureWorks
sample databases are based, is a large, multinational manufacturing company.
The company manufactures and sells metal and composite bicycles to North
American, European and Asian commercial markets. While its base operation is
located in Washington with 290 employees, several regional sales teams are
located throughout their market base.";
//Appends paragraph.
paragraph = section.AddParagraph();
textRange = paragraph.AppendText(text) as WTextRange;
paragraph = section.AddParagraph();
textRange = paragraph.AppendText(text) as WTextRange;
//Add image to the paragraph
paragraph = section.AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
FileStream imageStream = new FileStream(@"Mountain-200.jpg", FileMode.Open,
FileAccess.Read);
paragraph.AppendPicture(imageStream);
//Appends paragraph.
paragraph = section.AddParagraph();
textRange = paragraph.AppendText(text) as WTextRange;
//Instantiation of DocIORenderer for Word to PDF conversion
```



```

DocIORenderer render = new DocIORenderer();
//Converts Word document into PDF document
PdfDocument pdfDocument = render.ConvertToPDF(document);
//Releases all resources used by the Word document and DocIO Renderer
objects
render.Dispose();
document.Dispose();
//Save the document into stream.
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Close the documents.
pdfDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//A new document is created.
WordDocument document = new WordDocument();
//Adding a new section to the document.
WSection section = document.AddSection() as WSection;
//Set Margin of the section
section.PageSetup.Margins.All = 20;
//Create Paragraph styles
WParagraphStyle style = document.AddParagraphStyle("Normal") as
WParagraphStyle;
style.CharacterFormat.FontName = "Calibri";
style.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Justify;
style.CharacterFormat.FontSize = 11f;
style.ParagraphFormat.AfterSpacing = 8;
style.ParagraphFormat.FirstLineIndent = 36f;
style = document.AddParagraphStyle("Heading 1") as WParagraphStyle;
style.ApplyBaseStyle("Normal");
style.CharacterFormat.FontName = "Calibri Light";
style.CharacterFormat.FontSize = 16f;
style.CharacterFormat.TextColor = Syncfusion.Drawing.Color.FromArgb(46, 116,
181);
//Appends paragraph.
IWParagraph paragraph = section.AddParagraph();
paragraph.ApplyStyle("Heading 1");
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
paragraph.ParagraphFormat.AfterSpacing = 10;
WTextRange textRange = paragraph.AppendText("Adventure Works Cycles") as
WTextRange;
textRange.CharacterFormat.FontSize = 18f;
textRange.CharacterFormat.FontName = "Calibri";
string text =

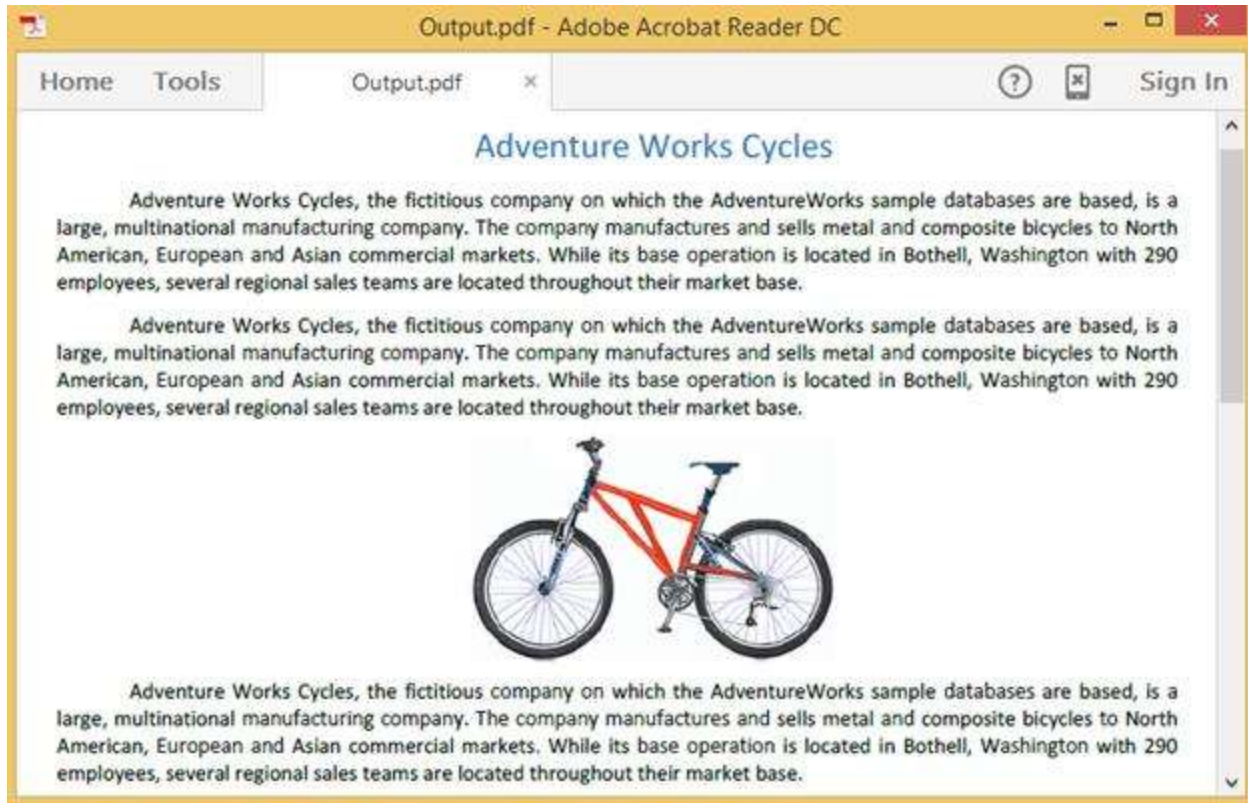
```

```

"Adventure Works Cycles, the fictitious company on which the AdventureWorks
sample databases are based, is a large, multinational manufacturing company.
The company manufactures and sells metal and composite bicycles to North
American, European and Asian commercial markets. While its base operation is
located in Washington with 290 employees, several regional sales teams are
located throughout their market base.";
//Appends paragraph.
paragraph = section.AddParagraph();
textRange = paragraph.AppendText(text) as WTextRange;
paragraph = section.AddParagraph();
textRange = paragraph.AppendText(text) as WTextRange;
//Add image to the paragraph
paragraph = section.AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DocIO.DLS.HorizontalAlignment.Center;
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Mountain-200.jpg");
paragraph.AppendPicture(imageStream);
//Appends paragraph.
paragraph = section.AddParagraph();
textRange = paragraph.AppendText(text) as WTextRange;
//Instantiation of DocIORenderer for Word to PDF conversion
DocIORenderer render = new DocIORenderer();
//Converts Word document into PDF document
PdfDocument pdfDocument = render.ConvertToPDF(document);
//Releases all resources used by the Word document and DocIO Renderer
objects
render.Dispose();
document.Dispose();
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Close the document
pdfDocument.Close();
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following image shows the PDF document with multiple paragraphs and image created using the flow model.



Working with table

You can create PDF document with simple table using the following code snippet.

C#

```
//Creates a new Word document
WordDocument wordDocument = new WordDocument();
//Adding a new section to the document.
WSection section = wordDocument.AddSection() as WSection;
//Set Margin of the section
section.PageSetup.Margins.All = 20;
// Adding a new Table
WTable table = section.AddTable() as WTable;
table.TableFormat.Paddings.All = 2;
table.TableFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Single;
// Inserting rows to the table.
table.ResetCells(4, 5);
//Appends paragraph with header.
IWParagraph paragraph = table[0, 0].AddParagraph();
paragraph.AppendText("SNO");
paragraph = table[0, 1].AddParagraph();
paragraph.AppendText("PRODUCT");
paragraph = table[0, 2].AddParagraph();
paragraph.AppendText("PRICE ($)");
paragraph = table[0, 3].AddParagraph();
paragraph.AppendText("QUANTITY");
paragraph = table[0, 4].AddParagraph();
paragraph.AppendText("TOTAL PRICE ($)");
//Add the first item
```

```

paragraph = table[1, 0].AddParagraph();
paragraph.AppendText("1");
paragraph = table[1, 1].AddParagraph();
paragraph.AppendText("AWC Logo Cap");
paragraph = table[1, 2].AddParagraph();
paragraph.AppendText("8.99");
paragraph = table[1, 3].AddParagraph();
paragraph.AppendText("2");
paragraph = table[1, 4].AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
paragraph.AppendText("17.98");
//Add the second item
paragraph = table[2, 0].AddParagraph();
paragraph.AppendText("2");
paragraph = table[2, 1].AddParagraph();
paragraph.AppendText("Long-Sleeve Logo Jersey, M");
paragraph = table[2, 2].AddParagraph();
paragraph.AppendText("49.99");
paragraph = table[2, 3].AddParagraph();
paragraph.AppendText("3");
paragraph = table[2, 4].AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
paragraph.AppendText("149.97");
// Table formatting with cell merging.
table[3,0].CellFormat.HorizontalMerge = CellMerge.Start;
table[3, 1].CellFormat.HorizontalMerge = CellMerge.Continue;
table[3, 2].CellFormat.HorizontalMerge = CellMerge.Continue;
table[3, 3].CellFormat.HorizontalMerge = CellMerge.Continue;
paragraph = table[3, 0].AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
paragraph.AppendText("Grand Total");
paragraph = table[3, 4].AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
paragraph.AppendText("167.95");
//Apply built-in table style to the table.
table.ApplyStyle(BuiltinTableStyle.MediumShading1Accent1);
//Creates an instance of the DocToPDFConverter
DocToPDFConverter converter = new DocToPDFConverter();
//Converts Word document into PDF document
PdfDocument pdfDocument = converter.ConvertToPDF(wordDocument);
//Save and close the PDF document
pdfDocument.Save("Output.pdf");
pdfDocument.Close(true);
//Close the document
wordDocument.Close();

```

VB.NET

```

'Creates a new Word document
Dim wordDocument As New WordDocument()
'Adding a new section to the document.
Dim section As WSection = TryCast(wordDocument.AddSection(), WSection)
'Set Margin of the section
section.PageSetup.Margins.All = 20
' Adding a new Table
Dim table As WTable = TryCast(section.AddTable(), WTable)

```

```

table.TableFormat.Paddings.All = 2
table.TableFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.[Single]
' Inserting rows to the table.
table.ResetCells(4, 5)
'Appends paragraph with header.
Dim paragraph As IWPParagraph = table(0, 0).AddParagraph()
paragraph.AppendText("SNO")
paragraph = table(0, 1).AddParagraph()
paragraph.AppendText("PRODUCT")
paragraph = table(0, 2).AddParagraph()
paragraph.AppendText("PRICE ($)")
paragraph = table(0, 3).AddParagraph()
paragraph.AppendText("QUANTITY")
paragraph = table(0, 4).AddParagraph()
paragraph.AppendText("TOTAL PRICE ($)")
'Add the first item
paragraph = table(1, 0).AddParagraph()
paragraph.AppendText("1")
paragraph = table(1, 1).AddParagraph()
paragraph.AppendText("AWC Logo Cap")
paragraph = table(1, 2).AddParagraph()
paragraph.AppendText("8.99")
paragraph = table(1, 3).AddParagraph()
paragraph.AppendText("2")
paragraph = table(1, 4).AddParagraph()
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right
paragraph.AppendText("17.98")
'Add the second item
paragraph = table(2, 0).AddParagraph()
paragraph.AppendText("2")
paragraph = table(2, 1).AddParagraph()
paragraph.AppendText("Long-Sleeve Logo Jersey, M")
paragraph = table(2, 2).AddParagraph()
paragraph.AppendText("49.99")
paragraph = table(2, 3).AddParagraph()
paragraph.AppendText("3")
paragraph = table(2, 4).AddParagraph()
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right
paragraph.AppendText("149.97")
' Table formatting with cell merging.
table(3, 0).CellFormat.HorizontalMerge = CellMerge.Start
table(3, 1).CellFormat.HorizontalMerge = CellMerge.[Continue]
table(3, 2).CellFormat.HorizontalMerge = CellMerge.[Continue]
table(3, 3).CellFormat.HorizontalMerge = CellMerge.[Continue]
paragraph = table(3, 0).AddParagraph()
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right
paragraph.AppendText("Grand Total")
paragraph = table(3, 4).AddParagraph()
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right
paragraph.AppendText("167.95")
'Apply built-in table style to the table.
table.ApplyStyle(BuiltinTableStyle.MediumShading1Accent1)
'Creates an instance of the DocToPDFConverter
Dim converter As New DocToPDFConverter()
'Converts Word document into PDF document
Dim pdfDocument As PdfDocument = converter.ConvertToPDF(wordDocument)

```

```
'Save and close the PDF document
pdfDocument.Save("Output.pdf")
pdfDocument.Close(True)
'Close the document
wordDocument.Close()
```

ASP.NET CORE

```
//Creates a new Word document
WordDocument wordDocument = new WordDocument();
//Adding a new section to the document
WSection section = wordDocument.AddSection() as WSection;
//Set Margin of the section
section.PageSetup.Margins.All = 20;
// Adding a new Table
WTable table = section.AddTable() as WTable;
table.TableFormat.Paddings.All = 2;
table.TableFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Single;
// Inserting rows to the table.
table.ResetCells(4, 5);
//Appends paragraph with header
IWParagraph paragraph = table[0, 0].AddParagraph();
paragraph.AppendText("SNO");
paragraph = table[0, 1].AddParagraph();
paragraph.AppendText("PRODUCT");
paragraph = table[0, 2].AddParagraph();
paragraph.AppendText("PRICE ($)");
paragraph = table[0, 3].AddParagraph();
paragraph.AppendText("QUANTITY");
paragraph = table[0, 4].AddParagraph();
paragraph.AppendText("TOTAL PRICE ($)");
//Add the first item
paragraph = table[1, 0].AddParagraph();
paragraph.AppendText("1");
paragraph = table[1, 1].AddParagraph();
paragraph.AppendText("AWC Logo Cap");
paragraph = table[1, 2].AddParagraph();
paragraph.AppendText("8.99");
paragraph = table[1, 3].AddParagraph();
paragraph.AppendText("2");
paragraph = table[1, 4].AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
paragraph.AppendText("17.98");
//Add the second item
paragraph = table[2, 0].AddParagraph();
paragraph.AppendText("2");
paragraph = table[2, 1].AddParagraph();
paragraph.AppendText("Long-Sleeve Logo Jersey, M");
paragraph = table[2, 2].AddParagraph();
paragraph.AppendText("49.99");
paragraph = table[2, 3].AddParagraph();
paragraph.AppendText("3");
paragraph = table[2, 4].AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
paragraph.AppendText("149.97");
```

```
// Table formatting with cell merging.
table[3, 0].CellFormat.HorizontalMerge = CellMerge.Start;
table[3, 1].CellFormat.HorizontalMerge = CellMerge.Continue;
table[3, 2].CellFormat.HorizontalMerge = CellMerge.Continue;
table[3, 3].CellFormat.HorizontalMerge = CellMerge.Continue;
paragraph = table[3, 0].AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
paragraph.AppendText("Grand Total");
paragraph = table[3, 4].AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
paragraph.AppendText("167.95");
//Apply built-in table style to the table
table.ApplyStyle(BuiltInTableStyle.MediumShading1Accent1);
//Instantiation of DocIORenderer for Word to PDF conversion
DocIORenderer render = new DocIORenderer();
//Converts Word document into PDF document
PdfDocument pdfDocument = render.ConvertToPDF(wordDocument);
//Releases all resources used by the Word document and DocIO Renderer
objects
render.Dispose();
wordDocument.Dispose();
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Close the documents
pdfDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = " Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new Word document
WordDocument wordDocument = new WordDocument();
//Adding a new section to the document.
WSection section = wordDocument.AddSection() as WSection;
//Set Margin of the section
section.PageSetup.Margins.All = 20;
// Adding a new Table
WTable table = section.AddTable() as WTable;
table.TableFormat.Paddings.All = 2;
table.TableFormat.Borders.BorderType =
Syncfusion.DocIO.DLS.BorderStyle.Single;
// Inserting rows to the table.
table.ResetCells(4, 5);
//Appends paragraph with header.
IWParagraph paragraph = table[0, 0].AddParagraph();
paragraph.AppendText("SNO");
paragraph = table[0, 1].AddParagraph();
paragraph.AppendText("PRODUCT");
paragraph = table[0, 2].AddParagraph();
```

```

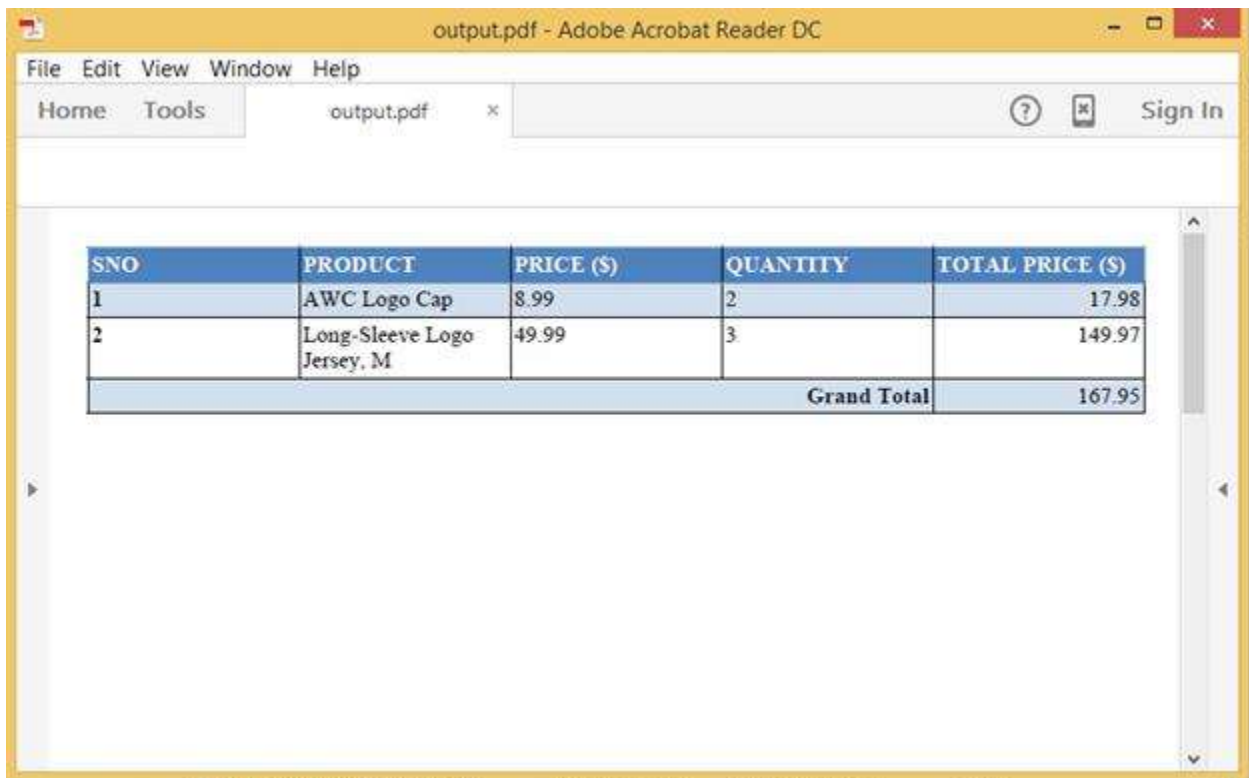
paragraph.AppendText("PRICE ($)");
paragraph = table[0, 3].AddParagraph();
paragraph.AppendText("QUANTITY");
paragraph = table[0, 4].AddParagraph();
paragraph.AppendText("TOTAL PRICE ($)");
//Add the first item
paragraph = table[1, 0].AddParagraph();
paragraph.AppendText("1");
paragraph = table[1, 1].AddParagraph();
paragraph.AppendText("AWC Logo Cap");
paragraph = table[1, 2].AddParagraph();
paragraph.AppendText("8.99");
paragraph = table[1, 3].AddParagraph();
paragraph.AppendText("2");
paragraph = table[1, 4].AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
paragraph.AppendText("17.98");
//Add the second item
paragraph = table[2, 0].AddParagraph();
paragraph.AppendText("2");
paragraph = table[2, 1].AddParagraph();
paragraph.AppendText("Long-Sleeve Logo Jersey, M");
paragraph = table[2, 2].AddParagraph();
paragraph.AppendText("49.99");
paragraph = table[2, 3].AddParagraph();
paragraph.AppendText("3");
paragraph = table[2, 4].AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
paragraph.AppendText("149.97");
// Table formatting with cell merging.
table[3, 0].CellFormat.HorizontalMerge = CellMerge.Start;
table[3, 1].CellFormat.HorizontalMerge = CellMerge.Continue;
table[3, 2].CellFormat.HorizontalMerge = CellMerge.Continue;
table[3, 3].CellFormat.HorizontalMerge = CellMerge.Continue;
paragraph = table[3, 0].AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
paragraph.AppendText("Grand Total");
paragraph = table[3, 4].AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment = HorizontalAlignment.Right;
paragraph.AppendText("167.95");
//Apply built-in table style to the table.
table.ApplyStyle(BuiltInTableStyle.MediumShading1Accent1);
//Instantiation of DocIORenderer for Word to PDF conversion
DocIORenderer render = new DocIORenderer();
//Converts Word document into PDF document
PdfDocument pdfDocument = render.ConvertToPDF(wordDocument);
//Releases all resources used by the Word document and DocIO Renderer
objects
render.Dispose();
wordDocument.Dispose();
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Close the document
pdfDocument.Close();
//Save the stream into pdf file

```



```
//The operation in Save under Xamarin varies between Windows Phone, Android
//and iOS platforms. Please refer PDF/Xamarin section for respective code
//samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

The following image shows the PDF document with simple table created using the flow model.



SNO	PRODUCT	PRICE (\$)	QUANTITY	TOTAL PRICE (\$)
1	AWC Logo Cap	8.99	2	17.98
2	Long-Sleeve Logo Jersey, M	49.99	3	149.97
Grand Total				167.95

Flow model using PdfLayoutResult

Syncfusion Essential PDF supports creating a PDF document with flow model by maintaining the position of previously drawn element in [PdfLayoutResult](#).

The following code snippet explains how to create a PDF document with image, paragraph text, header text, a line below the header text, and a table using flow model.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
```

```

//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load an image into PdfImage
PdfImage image = PdfImage.FromFile("AdventureCycle.jpg");
//Draw image on the page in the specified location and with required size
graphics.DrawImage(image, new RectangleF(150, 30, 200, 100));
//Initialize a standard font
PdfFont standardFont = new PdfStandardFont(PdfFontFamily.Helvetica, 12);
//Load the paragraph text into PdfTextElement with initialized standard font
PdfTextElement textElement = new PdfTextElement("Adventure Works Cycles, the
fictitious company on which the AdventureWorks sample databases are based,"
+
" is a large, multinational manufacturing company. The company manufactures
and sells metal and composite bicycles to North American, " +
"European and Asian commercial markets. While its base operation is located
in Bothell, Washington with 290 employees, several regional" +
" sales teams are located throughout their market base.", standardFont);
//Draw the paragraph text on page and maintain the position in
PdfLayoutResult
PdfLayoutResult layoutResult = textElement.Draw(page, new RectangleF(0, 150,
page.GetClientSize().Width, page.GetClientSize().Height));
//Assign header text to PdfTextElement
textElement.Text = "Top 5 sales stores";
//Assign standard font to PdfTextElement
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Draw the header text on page, below the paragraph text with a height gap
of 20 and maintain the position in PdfLayoutResult
layoutResult = textElement.Draw(page, new PointF(0,
layoutResult.Bounds.Bottom + 20));
//Initialize PdfLine with start point and end point for drawing the line
PdfLine line = new PdfLine(new PointF(0, 0), new
PointF(page.GetClientSize().Width, 0))
{
    Pen = PdfPens.DarkGray
};
//Draw the line on page, below the header text with a height gap of 5 and
maintain the position in PdfLayoutResult
layoutResult = line.Draw(page, new PointF(0, layoutResult.Bounds.Bottom +
5));
//Initialize PdfGrid for drawing the table
PdfGrid grid = new PdfGrid();
//Create a DataTable
DataTable dataTable = new DataTable();
//Add columns to the DataTable
dataTable.Columns.Add("ID");
dataTable.Columns.Add("Name");
dataTable.Columns.Add("Salary");
//Add rows to the DataTable
dataTable.Rows.Add(new object[] { "E01", "Clay", "$10,000" });
dataTable.Rows.Add(new object[] { "E02", "Thomas", "$10,500" });
dataTable.Rows.Add(new object[] { "E03", "Simon", "$12,000" });
//Assign the DataTable as data source to grid
grid.DataSource = dataTable;
//Set the grid cell padding
grid.Style.CellPadding.All = 5;
//Apply built-in table style to the grid

```

```

grid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable5DarkAccent5);
//Draw the table in page, below the line with a height gap of 20
grid.Draw(page, new PointF(0, layoutResult.Bounds.Bottom + 20));
//Save the PDF document
document.Save("Output.pdf");
//Close the instance of PdfDocument
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = document.Pages.Add
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Load an image into PdfImage
Dim image As PdfImage = PdfImage.FromFile("AdventureCycle.jpg")
'Draw image on the page in the specified location and with required size
graphics.DrawImage(image, New RectangleF(150, 30, 200, 100))
'Initialize a standard font
Dim standardFont As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica,
12)
'Load the paragraph text into PdfTextElement with initialized standard font
Dim textElement As PdfTextElement = New PdfTextElement("Adventure Works
Cycles, the fictitious company on which the AdventureWorks sample databases
are based," &
" is a large, multinational manufacturing company. The company manufactures
and sells metal and composite bicycles to North American, " &
"European And Asian commercial markets. While its base operation Is located
in Bothell, Washington with 290 employees, several regional" &
" sales teams are located throughout their market base.", standardFont)
'Draw the paragraph text on page and maintain the position in
PdfLayoutResult
Dim layoutResult As PdfLayoutResult = textElement.Draw(page, New
RectangleF(0, 150, page.GetClientSize.Width, page.GetClientSize.Height))
'Assign header text to PdfTextElement
textElement.Text = "Top 5 sales stores"
'Assign standard font to PdfTextElement
textElement.Font = New PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold)
'Draw the header text on page, below the paragraph text with a height gap of
20 and maintain the position in PdfLayoutResult
layoutResult = textElement.Draw(page, New PointF(0,
(layoutResult.Bounds.Bottom + 20)))
'Initialize PdfLine with start point and end point for drawing the line
Dim line As PdfLine = New PdfLine(New PointF(0, 0), New
PointF(page.GetClientSize.Width, 0))
'Draw the line on page, below the header text with a height gap of 5 and
maintain the position in PdfLayoutResult
layoutResult = line.Draw(page, New PointF(0, (layoutResult.Bounds.Bottom +
5)))
'Initialize PdfGrid for drawing the table
Dim grid As PdfGrid = New PdfGrid
'Create a DataTable
Dim dataTable As DataTable = New DataTable

```

```

'Add columns to the DataTable
dataTable.Columns.Add("ID")
dataTable.Columns.Add("Name")
dataTable.Columns.Add("Salary")
'Add rows to the DataTable
dataTable.Rows.Add(New Object() {"E01", "Clay", "$10,000"})
dataTable.Rows.Add(New Object() {"E02", "Thomas", "$10,500"})
dataTable.Rows.Add(New Object() {"E03", "Simon", "$12,000"})
'Assign the DataTable as data source to grid
grid.DataSource = dataTable
'Set the grid cell padding
grid.Style.CellPadding.All = 5
'Apply built-in table style to the grid
grid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable5DarkAccent5)
'Draw the table in page, below the line with a height gap of 20
grid.Draw(page, New PointF(0, (layoutResult.Bounds.Bottom + 20)))
'Save the PDF document
document.Save("Output.pdf")
'Close the instance of PdfDocument
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load an image into PdfImage
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.AdventureCycle.jpg");
PdfImage image = PdfImage.FromStream(imageStream);
//Draw image on the page in the specified location and with required size
graphics.DrawImage(image, new RectangleF(150, 30, 200, 100));
//Initialize a standard font
PdfFont standardFont = new PdfStandardFont(PdfFontFamily.Helvetica, 12);
//Load the paragraph text into PdfTextElement with initialized standard font
PdfTextElement textElement = new PdfTextElement("Adventure Works Cycles, the
fictitious company on which the AdventureWorks sample databases are based,"
+
" is a large, multinational manufacturing company. The company manufactures
and sells metal and composite bicycles to North American, " +
"European and Asian commercial markets. While its base operation is located
in Bothell, Washington with 290 employees, several regional" +
" sales teams are located throughout their market base.", standardFont);
//Draw the paragraph text on page and maintain the position in
PdfLayoutResult
PdfLayoutResult layoutResult = textElement.Draw(page, new RectangleF(0, 150,
page.GetClientSize().Width, page.GetClientSize().Height));
//Assign header text to PdfTextElement
textElement.Text = "Top 5 sales stores";
//Assign standard font to PdfTextElement
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);

```

```

//Draw the header text on page, below the paragraph text with a height gap
of 20 and maintain the position in PdfLayoutResult
layoutResult = textElement.Draw(page, new PointF(0,
layoutResult.Bounds.Bottom + 20));
//Initialize PdfLine with start point and end point for drawing the line
PdfLine line = new PdfLine(new PointF(0, 0), new
PointF(page.ClientSize().Width, 0))
{
    Pen = PdfPens.DarkGray
};
//Draw the line on page, below the header text with a height gap of 5 and
maintain the position in PdfLayoutResult
layoutResult = line.Draw(page, new PointF(0, layoutResult.Bounds.Bottom +
5));
//Initialize PdfGrid for drawing the table
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridlrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridlrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridlrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridlrow1);
data.Add(gridlrow2);
data.Add(gridlrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign the DataTable as data source to grid
grid.DataSource = dataTable;
//Set the grid cell padding
grid.Style.CellPadding.All = 5;
//Apply built-in table style to the grid
grid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable5DarkAccent5);
//Draw the table in page, below the line with a height gap of 20
grid.Draw(page, new PointF(0, layoutResult.Bounds.Bottom + 20));
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load an image into PdfImage
FileStream imageStream = new FileStream("AdventureCycle.jpg",
    FileMode.Open);
PdfImage image = PdfImage.FromStream(imageStream);

```

```

//Draw image on the page in the specified location and with required size
graphics.DrawImage(image, new RectangleF(150, 30, 200, 100));
//Initialize a standard font
PdfFont standardFont = new PdfStandardFont(PdfFontFamily.Helvetica, 12);
//Load the paragraph text into PdfTextElement with initialized standard font
PdfTextElement textElement = new PdfTextElement("Adventure Works Cycles, the
fictitious company on which the AdventureWorks sample databases are based,"
+
" is a large, multinational manufacturing company. The company manufactures
and sells metal and composite bicycles to North American, " +
"European and Asian commercial markets. While its base operation is located
in Bothell, Washington with 290 employees, several regional" +
" sales teams are located throughout their market base.", standardFont);
//Draw the paragraph text on page and maintain the position in
PdfLayoutResult
PdfLayoutResult layoutResult = textElement.Draw(page, new RectangleF(0, 150,
page.GetClientSize().Width, page.GetClientSize().Height));
//Assign header text to PdfTextElement
textElement.Text = "Top 5 sales stores";
//Assign standard font to PdfTextElement
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Draw the header text on page, below the paragraph text with a height gap
of 20 and maintain the position in PdfLayoutResult
layoutResult = textElement.Draw(page, new PointF(0,
layoutResult.Bounds.Bottom + 20));
//Initialize PdfLine with start point and end point for drawing the line
PdfLine line = new PdfLine(new PointF(0, 0), new
PointF(page.GetClientSize().Width, 0))
{
    Pen = PdfPens.DarkGray
};
//Draw the line on page, below the header text with a height gap of 5 and
maintain the position in PdfLayoutResult
layoutResult = line.Draw(page, new PointF(0, layoutResult.Bounds.Bottom +
5));
//Initialize PdfGrid for drawing the table
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridlrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridlrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridlrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridlrow1);
data.Add(gridlrow2);
data.Add(gridlrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign the DataTable as data source to grid
grid.DataSource = dataTable;
//Set the grid cell padding
grid.Style.CellPadding.All = 5;
//Apply built-in table style to the grid
grid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable5DarkAccent5);
//Draw the table in page, below the line with a height gap of 20
grid.Draw(page, new PointF(0, layoutResult.Bounds.Bottom + 20));
//Saving the PDF to the MemoryStream

```

```

MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownloadName = "Output.pdf";
return fileStreamResult;

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load an image into PdfImage
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.AdventureCycle.jpg");
PdfImage image = PdfImage.FromStream(imageStream);
//Draw image on the page in the specified location and with required size
graphics.DrawImage(image, new RectangleF(150, 30, 200, 100));
//Initialize a standard font
PdfFont standardFont = new PdfStandardFont(PdfFontFamily.Helvetica, 12);
//Load the paragraph text into PdfTextElement with initialized standard font
PdfTextElement textElement = new PdfTextElement("Adventure Works Cycles, the
fictitious company on which the AdventureWorks sample databases are based,"
+
" is a large, multinational manufacturing company. The company manufactures
and sells metal and composite bicycles to North American, " +
"European and Asian commercial markets. While its base operation is located
in Bothell, Washington with 290 employees, several regional" +
" sales teams are located throughout their market base.", standardFont);
//Draw the paragraph text on page and maintain the position in
PdfLayoutResult
PdfLayoutResult layoutResult = textElement.Draw(page, new RectangleF(0, 150,
page.GetClientSize().Width, page.GetClientSize().Height));
//Assign header text to PdfTextElement
textElement.Text = "Top 5 sales stores";
//Assign standard font to PdfTextElement
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Draw the header text on page, below the paragraph text with a height gap
of 20 and maintain the position in PdfLayoutResult
layoutResult = textElement.Draw(page, new PointF(0,
layoutResult.Bounds.Bottom + 20));
//Initialize PdfLine with start point and end point for drawing the line
PdfLine line = new PdfLine(new PointF(0, 0), new
PointF(page.GetClientSize().Width, 0))
{
    Pen = PdfPens.DarkGray
};

```

```

//Draw the line on page, below the header text with a height gap of 5 and maintain the position in PdfLayoutResult
layoutResult = line.Draw(page, new PointF(0, layoutResult.Bounds.Bottom + 5));
//Initialize PdfGrid for drawing the table
PdfGrid grid = new PdfGrid();
//Add values to list
List<object> data = new List<object>();
Object gridlrow1 = new { ID = "E01", Name = "Clay", Salary = "$10,000" };
Object gridlrow2 = new { ID = "E02", Name = "Thomas", Salary = "$10,500" };
Object gridlrow3 = new { ID = "E03", Name = "Simon", Salary = "$12,000" };
data.Add(gridlrow1);
data.Add(gridlrow2);
data.Add(gridlrow3);
//Add list to IEnumerable
IEnumerable<object> dataTable = data;
//Assign the DataTable as data source to grid
grid.DataSource = dataTable;
//Set the grid cell padding
grid.Style.CellPadding.All = 5;
//Apply built-in table style to the grid
grid.ApplyBuiltinStyle(PdfGridBuiltinStyle.GridTable5DarkAccent5);
//Draw the table in page, below the line with a height gap of 20
grid.Draw(page, new PointF(0, layoutResult.Bounds.Bottom + 20));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android, and iOS platforms. Refer to the PDF/Xamarin section for respective code samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf", "application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf", "application/pdf", stream);
}

```

Working with Forms

An interactive form, sometimes referred to as an AcroForm is a collection of fields for gathering information. A PDF document can contain any number of fields appearing on any combination of pages, all of that make a single, globally interactive form spanning the entire document.

Creating a new PDF form

Essential PDF allows you to create and manage the form (AcroForm) in PDF document by using [PdfForm](#) class. The [PdfFormFieldCollection](#) class represents the entire field collection of the form.

Adding the text box field

[PdfTextBoxField](#) class is used to create a text box field in PDF forms.

The below code snippet illustrates how to add a textbox field to a new PDF document.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to the PDF document.
PdfPage page = document.Pages.Add();
//Create a textbox field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Add the form field to the document.
document.Form.Fields.Add(textBoxField);
//Save the document.
document.Save("Form.pdf");
//close the document
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As PdfDocument = New PdfDocument()
'Add a new page to the PDF document.
Dim page As PdfPage = document.Pages.Add()
'Create a textbox field and add the properties.
Dim textBoxField As PdfTextBoxField = New PdfTextBoxField(page, "FirstName")
textBoxField.Bounds = New RectangleF(0, 0, 100, 20)
textBoxField.ToolTip = "First Name"
'Add the form field to the document.
document.Form.Fields.Add(textBoxField)
'Save the document.
document.Save("Form.pdf")
'close the document
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to the PDF document.
PdfPage page = document.Pages.Add();
//Create a textbox field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Add the form field to the document.
document.Form.Fields.Add(textBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
```

```
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Form.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to the PDF document.
PdfPage page = document.Pages.Add();
//Create a textbox field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Add the form field to the document.
document.Form.Fields.Add(textBoxField);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to the PDF document.
PdfPage page = document.Pages.Add();
//Create a textbox field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Add the form field to the document.
document.Form.Fields.Add(textBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}
```

The below code snippet illustrates how to add the textbox to an existing PDF document.

C#

```
//Load the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Create the form if the form does not exist in the loaded document
if(loadedDocument.Form==null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a textbox field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(loadedPage, "FirstName");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Add the form field to the existing PDF document.
loadedDocument.Form.Fields.Add(textBoxField);
//Save the document.
loadedDocument.Save("Form.pdf");
//close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Create the form if the form does not exist in the loaded document
If loadedDocument.Form Is Nothing Then
loadedDocument.CreateForm()
End If
'Load the page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Create a text box field and add the properties.
Dim textBoxField As New PdfTextBoxField(loadedPage, "FirstName")
textBoxField.Bounds = New RectangleF(0, 0, 100, 20)
textBoxField.ToolTip = "First Name"
'Add the form field to the existing PDF document.
loadedDocument.Form.Fields.Add(textBoxField)
'Save the document.
loadedDocument.Save("Form.pdf")
'close the document
loadedDocument.Close(True)
```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a textbox field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(loadedPage, "FirstName");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Add the form field to the existing PDF document.
loadedDocument.Form.Fields.Add(textBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples.
Save(stream, "Form.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a textbox field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(loadedPage, "FirstName");
textBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Add the form field to the existing PDF document.
loadedDocument.Form.Fields.Add(textBoxField);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.

```

```
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a textbox field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(loadedPage, "FirstName");
textBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Add the form field to the existing PDF document.
loadedDocument.Form.Fields.Add(textBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}
```

Adding the combo box field

[PdfComboBoxField](#) class is used to create a combo box field in PDF forms. You can add a list of items to the combo box by using the [PdfListFieldItem](#) class.

Please refer the below code snippet for adding the combo box in new PDF document.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
```

```

//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a combo box for the first page.
PdfComboBoxField comboBoxField = new PdfComboBoxField(page, "JobTitle");
//Set the combo box properties.
comboBoxField.Bounds = new RectangleF(0, 40, 100, 20);
//Set tooltip.
comboBoxField.ToolTip = "Job Title";
//Add list items.
comboBoxField.Items.Add(new PdfListFieldItem("Development", "accounts"));
comboBoxField.Items.Add(new PdfListFieldItem("Support", "advertise"));
comboBoxField.Items.Add(new PdfListFieldItem("Documentation", "content"));
//Add combo box to the form.
document.Form.Fields.Add(comboBoxField);
//Save the document.
document.Save("Form.pdf");
//Close the document.
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
'Add a new page to PDF document.
Dim page As PdfPage = document.Pages.Add()
'Create a combo box for the first page.
Dim comboBoxField As New PdfComboBoxField(page, "JobTitle")
'Set the combo box properties.
comboBoxField.Bounds = New RectangleF(0, 40, 100, 20)
'Set tooltip
comboBoxField.ToolTip = "Job Title"
'Add list items.
comboBoxField.Items.Add(New PdfListFieldItem("Development", "accounts"))
comboBoxField.Items.Add(New PdfListFieldItem("Support", "advertise"))
comboBoxField.Items.Add(New PdfListFieldItem("Documentation", "content"))
'Add combo box to the form.
document.Form.Fields.Add(comboBoxField)
'Save the PDF document.
document.Save("Form.pdf")
'Close the document
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a combo box for the first page.
PdfComboBoxField comboBoxField = new PdfComboBoxField(page, "JobTitle");
//Set the combo box properties.
comboBoxField.Bounds = new RectangleF(0, 40, 100, 20);
//Set tooltip.
comboBoxField.ToolTip = "Job Title";
//Add list items.
comboBoxField.Items.Add(new PdfListFieldItem("Development", "accounts"));

```

```
comboBoxField.Items.Add(new PdfListFieldItem("Support", "advertise"));
comboBoxField.Items.Add(new PdfListFieldItem("Documentation", "content"));
//Add combo box to the form.
document.Form.Fields.Add(comboBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Form.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a combo box for the first page.
PdfComboBoxField comboBoxField = new PdfComboBoxField(page, "JobTitle");
//Set the combo box properties.
comboBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 40, 100, 20);
//Set tooltip.
comboBoxField.ToolTip = "Job Title";
//Add list items.
comboBoxField.Items.Add(new PdfListFieldItem("Development", "accounts"));
comboBoxField.Items.Add(new PdfListFieldItem("Support", "advertise"));
comboBoxField.Items.Add(new PdfListFieldItem("Documentation", "content"));
//Add combo box to the form.
document.Form.Fields.Add(comboBoxField);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a combo box for the first page.
PdfComboBoxField comboBoxField = new PdfComboBoxField(page, "JobTitle");
//Set the combo box properties.
```

```

comboBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 40, 100, 20);
//Set tooltip.
comboBoxField.ToolTip = "Job Title";
//Add list items.
comboBoxField.Items.Add(new PdfListFieldItem("Development", "accounts"));
comboBoxField.Items.Add(new PdfListFieldItem("Support", "advertise"));
comboBoxField.Items.Add(new PdfListFieldItem("Documentation", "content"));
//Add combo box to the form.
document.Form.Fields.Add(comboBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}

```

Please refer the below code snippet for adding the combo box in existing PDF document.

C#

```

//Load the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Create the form if the form does not exist in the loaded document
if(loadedDocument.Form==null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a combo box for the first page.
PdfComboBoxField comboBoxField = new PdfComboBoxField(loadedPage,
"JobTitle");
//Set the combo box properties.
comboBoxField.Bounds = new RectangleF(0, 40, 100, 20);
//Set tooltip.
comboBoxField.ToolTip = "Job Title";
//Add list items.
comboBoxField.Items.Add(new PdfListFieldItem("Development", "accounts"));
comboBoxField.Items.Add(new PdfListFieldItem("Support", "advertise"));
comboBoxField.Items.Add(new PdfListFieldItem("Documentation", "content"));
//Add combo box to the form.
loadedDocument.Form.Fields.Add(comboBoxField);
//Save the document.
loadedDocument.Save("Form.pdf");

```



```
//Close the document.
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Create the form if the form does not exist in the loaded document
If loadedDocument.Form Is Nothing Then
loadedDocument.CreateForm()
End If
'Load the page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Create a combo box for the first page.
Dim comboBoxField As New PdfComboBoxField(loadedPage, "JobTitle")
'Set the combo box properties.
comboBoxField.Bounds = New RectangleF(0, 40, 100, 20)
'Set tooltip.
comboBoxField.ToolTip = "Job Title"
'Add list items.
comboBoxField.Items.Add(New PdfListFieldItem("Development", "accounts"))
comboBoxField.Items.Add(New PdfListFieldItem("Support", "advertise"))
comboBoxField.Items.Add(New PdfListFieldItem("Documentation", "content"))
'Add combo box to the form.
loadedDocument.Form.Fields.Add(comboBoxField)
'Save the document.
loadedDocument.Save("Form.pdf")
'Close the document.
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a combo box for the first page.
PdfComboBoxField comboBoxField = new PdfComboBoxField(loadedPage,
"JobTitle");
//Set the combo box properties.
comboBoxField.Bounds = new RectangleF(0, 40, 100, 20);
//Set tooltip.
comboBoxField.ToolTip = "Job Title";
//Add list items.
```

```

comboBoxField.Items.Add(new PdfListFieldItem("Development", "accounts"));
comboBoxField.Items.Add(new PdfListFieldItem("Support", "advertise"));
comboBoxField.Items.Add(new PdfListFieldItem("Documentation", "content"));
//Add combo box to the form.
loadedDocument.Form.Fields.Add(comboBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Form.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a combo box for the first page.
PdfComboBoxField comboBoxField = new PdfComboBoxField(loadedPage,
"JobTitle");
//Set the combo box properties.
comboBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 40, 100, 20);
//Set tooltip.
comboBoxField.ToolTip = "Job Title";
//Add list items.
comboBoxField.Items.Add(new PdfListFieldItem("Development", "accounts"));
comboBoxField.Items.Add(new PdfListFieldItem("Support", "advertise"));
comboBoxField.Items.Add(new PdfListFieldItem("Documentation", "content"));
//Add combo box to the form.
loadedDocument.Form.Fields.Add(comboBoxField);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a combo box for the first page.
PdfComboBoxField comboBoxField = new PdfComboBoxField(loadedPage,
"JobTitle");
//Set the combo box properties.
comboBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 40, 100, 20);
//Set tooltip.
comboBoxField.ToolTip = "Job Title";
//Add list items.
comboBoxField.Items.Add(new PdfListFieldItem("Development", "accounts"));
comboBoxField.Items.Add(new PdfListFieldItem("Support", "advertise"));
comboBoxField.Items.Add(new PdfListFieldItem("Documentation", "content"));
//Add combo box to the form.
loadedDocument.Form.Fields.Add(comboBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}

```

Adding the radio button field

To create the radio button in the PDF forms, you can use [PdfRadioButtonListField](#) class and you can create the radio button list items by using the [PdfRadioButtonListItem](#) class.

Please refer the below code snippet for adding the radio button in new PDF document.

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();

```

```

//Create a Radio button.
PdfRadioButtonListField employeesRadioList = new
PdfRadioButtonListField(page, "employeesRadioList");
//Add the radio button into form
document.Form.Fields.Add(employeesRadioList);
//Create radio button items.
PdfRadioButtonListItem radioButtonItem1 = new PdfRadioButtonListItem("1-9");
radioButtonItem1.Bounds = new RectangleF(100, 140, 20, 20);
PdfRadioButtonListItem radioButtonItem2 = new PdfRadioButtonListItem("10-
49");
radioButtonItem2.Bounds = new RectangleF(100, 170, 20, 20);
//Add the items to radio button group.
employeesRadioList.Items.Add(radioButtonItem1);
employeesRadioList.Items.Add(radioButtonItem2);
//Save the document.
document.Save("Form.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
'Add a new page to PDF document.
Dim page As PdfPage = document.Pages.Add()
'Create a Radio button.
Dim employeesRadioList As New PdfRadioButtonListField(page,
"employeesRadioList")
'Add the radio button into form
document.Form.Fields.Add(employeesRadioList)
'Create radio button items.
Dim radioItem1 As New PdfRadioButtonListItem("1-9")
radioItem1.Bounds = New RectangleF(100, 140, 20, 20)
Dim radioItem2 As New PdfRadioButtonListItem("10-49")
radioItem2.Bounds = New RectangleF(100, 170, 20, 20)
'Add the items to radio button group.
employeesRadioList.Items.Add(radioItem1)
employeesRadioList.Items.Add(radioItem2)
'Save the PDF document.
document.Save("Form.pdf")
'close the document
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a Radio button.
PdfRadioButtonListField employeesRadioList = new
PdfRadioButtonListField(page, "employeesRadioList");
//Add the radio button into form
document.Form.Fields.Add(employeesRadioList);
//Create radio button items.
PdfRadioButtonListItem radioButtonItem1 = new PdfRadioButtonListItem("1-9");

```

```

radioButtonItem1.Bounds = new RectangleF(100, 140, 20, 20);
PdfRadioButtonListItem radioButtonItem2 = new PdfRadioButtonListItem("10-49");
radioButtonItem2.Bounds = new RectangleF(100, 170, 20, 20);
//Add the items to radio button group.
employeesRadioList.Items.Add(radioButtonItem1);
employeesRadioList.Items.Add(radioButtonItem2);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Form.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a Radio button.
PdfRadioButtonListField employeesRadioList = new
PdfRadioButtonListField(page, "employeesRadioList");
//Add the radio button into form
document.Form.Fields.Add(employeesRadioList);
//Create radio button items.
PdfRadioButtonListItem radioButtonItem1 = new PdfRadioButtonListItem("1-9");
radioButtonItem1.Bounds = new Syncfusion.Drawing.RectangleF(100, 140, 20,
20);
PdfRadioButtonListItem radioButtonItem2 = new PdfRadioButtonListItem("10-49");
radioButtonItem2.Bounds = new Syncfusion.Drawing.RectangleF(100, 170, 20,
20);
//Add the items to radio button group.
employeesRadioList.Items.Add(radioButtonItem1);
employeesRadioList.Items.Add(radioButtonItem2);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a Radio button.
PdfRadioButtonListField employeesRadioList = new
PdfRadioButtonListField(page, "employeesRadioList");
//Add the radio button into form
document.Form.Fields.Add(employeesRadioList);
//Create radio button items.
PdfRadioButtonListItem radioButtonItem1 = new PdfRadioButtonListItem("1-9");
radioButtonItem1.Bounds = new Syncfusion.Drawing.RectangleF(100, 140, 20,
20);
PdfRadioButtonListItem radioButtonItem2 = new PdfRadioButtonListItem("10-
49");
radioButtonItem2.Bounds = new Syncfusion.Drawing.RectangleF(100, 170, 20,
20);
//Add the items to radio button group.
employeesRadioList.Items.Add(radioButtonItem1);
employeesRadioList.Items.Add(radioButtonItem2);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}

```

The below code snippet illustrates how to add the radio button in existing PDF document.

C#

```

//Load the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a Radio button.
PdfRadioButtonListField employeesRadioList = new
PdfRadioButtonListField(loadedPage, "employeesRadioList");
//Add the radio button into loaded document

```

```
loadedDocument.Form.Fields.Add(employeesRadioList);
//Create radio button items.
PdfRadioButtonListItem radioButtonItem1 = new PdfRadioButtonListItem("1-9");
radioButtonItem1.Bounds = new RectangleF(100, 140, 20, 20);
PdfRadioButtonListItem radioButtonItem2 = new PdfRadioButtonListItem("10-
49");
radioButtonItem2.Bounds = new RectangleF(100, 170, 20, 20);
//Add the items to radio button group.
employeesRadioList.Items.Add(radioButtonItem1);
employeesRadioList.Items.Add(radioButtonItem2);
//Save the document.
loadedDocument.Save("Form.pdf");
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Create the form if the form does not exist in the loaded document
If loadedDocument.Form Is Nothing Then
loadedDocument.CreateForm()
End If
'Load the page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Create a Radio button.
Dim employeesRadioList As New PdfRadioButtonListField(loadedPage,
"employeesRadioList")
'Add the radio button into loaded document
loadedDocument.Form.Fields.Add(employeesRadioList)
'Create radio button items.
Dim radioButtonItem1 As New PdfRadioButtonListItem("1-9")
radioButtonItem1.Bounds = New RectangleF(100, 140, 20, 20)
Dim radioButtonItem2 As New PdfRadioButtonListItem("10-49")
radioButtonItem2.Bounds = New RectangleF(100, 170, 20, 20)
'Add the items to radio button group.
employeesRadioList.Items.Add(radioButtonItem1)
employeesRadioList.Items.Add(radioButtonItem2)
'Save the document.
loadedDocument.Save("Form.pdf")
'Close the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
```

```

//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a Radio button.
PdfRadioButtonListField employeesRadioList = new
PdfRadioButtonListField(loadedPage, "employeesRadioList");
//Add the radio button into loaded document
loadedDocument.Form.Fields.Add(employeesRadioList);
//Create radio button items.
PdfRadioButtonListItem radioButtonItem1 = new PdfRadioButtonListItem("1-9");
radioButtonItem1.Bounds = new RectangleF(100, 140, 20, 20);
PdfRadioButtonListItem radioButtonItem2 = new PdfRadioButtonListItem("10-
49");
radioButtonItem2.Bounds = new RectangleF(100, 170, 20, 20);
//Add the items to radio button group.
employeesRadioList.Items.Add(radioButtonItem1);
employeesRadioList.Items.Add(radioButtonItem2);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Form.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a Radio button.
PdfRadioButtonListField employeesRadioList = new
PdfRadioButtonListField(loadedPage, "employeesRadioList");
//Add the radio button into loaded document
loadedDocument.Form.Fields.Add(employeesRadioList);
//Create radio button items.
PdfRadioButtonListItem radioButtonItem1 = new PdfRadioButtonListItem("1-9");
radioButtonItem1.Bounds = new Syncfusion.Drawing.RectangleF(100, 140, 20,
20);
PdfRadioButtonListItem radioButtonItem2 = new PdfRadioButtonListItem("10-
49");
radioButtonItem2.Bounds = new Syncfusion.Drawing.RectangleF(100, 170, 20,
20);
//Add the items to radio button group.
employeesRadioList.Items.Add(radioButtonItem1);
employeesRadioList.Items.Add(radioButtonItem2);
//Creating the stream object

```



```

MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a Radio button.
PdfRadioButtonListField employeesRadioList = new
PdfRadioButtonListField(loadedPage, "employeesRadioList");
//Add the radio button into loaded document
loadedDocument.Form.Fields.Add(employeesRadioList);
//Create radio button items.
PdfRadioButtonListItem radioButtonItem1 = new PdfRadioButtonListItem("1-9");
radioButtonItem1.Bounds = new Syncfusion.Drawing.RectangleF(100, 140, 20,
20);
PdfRadioButtonListItem radioButtonItem2 = new PdfRadioButtonListItem("10-
49");
radioButtonItem2.Bounds = new Syncfusion.Drawing.RectangleF(100, 170, 20,
20);
//Add the items to radio button group.
employeesRadioList.Items.Add(radioButtonItem1);
employeesRadioList.Items.Add(radioButtonItem2);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}
```

Retrieving option values from acroform radio button

The Essential PDF supports retrieving option values from acroform radio button. The [OptionValue](#) property is used to get option values of [PdfLoadedRadioButtonItem](#) instance.

The following code example illustrates how to get option values from acroform radio button.

C#

```
//Load an existing document
PdfLoadedDocument doc = new PdfLoadedDocument("SourceForm.pdf");
//Gets the loaded form
PdfLoadedForm form = doc.Form;
//Set default appearance to false
form.SetDefaultAppearance(false);
//Gets the 'Gender' radio button field
PdfLoadedRadioButtonListField radioButtonField = form.Fields["Gender"] as
PdfLoadedRadioButtonListField;
//Select the item that contains option value as "Male"
foreach (PdfLoadedRadioButtonItem item in radioButtonField.Items)
{
//Gets an option value of the item
if (item.OptionValue == "Male")
{
item.Selected = true;
}
}
//Save and close the PDF document
doc.Save("Form.pdf");
doc.Close(true);
```

VB.NET

```
'Load an existing document
Dim doc As New PdfLoadedDocument("SourceForm.pdf")
'Gets the loaded form
Dim form As PdfLoadedForm = doc.Form
'Set default appearance to false
form.SetDefaultAppearance(False)
'Gets the 'Gender' radio button field
Dim radioButtonField As PdfLoadedRadioButtonListField =
TryCast(form.Fields("Gender"), PdfLoadedRadioButtonListField)
'Select the item that contains option value as "Male"
For Each item As PdfLoadedRadioButtonItem In radioButtonField.Items
'Gets an option value of the item
If item.OptionValue = "Male" Then
item.Selected = True
End If
```

Next

```
'Save and close the PDF document
doc.Save("Form.pdf")
doc.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument doc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through the Open method of
PdfLoadedDocument class
await doc.OpenAsync(file);
//Gets the loaded form
PdfLoadedForm form = doc.Form;
//Set default appearance to false
form.SetDefaultAppearance(false);
//Gets the 'Gender' radio button field
PdfLoadedRadioButtonListField radioButtonField = form.Fields["Gender"] as
PdfLoadedRadioButtonListField;
//Select the item that contains option value as "Male"
foreach (PdfLoadedRadioButtonItem item in radioButtonField.Items)
{
    //Gets an option value of the item
    if (item.OptionValue == "Male")
    {
        item.Selected = true;
    }
}
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Form.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
//Gets the loaded form
PdfLoadedForm form = doc.Form;
//Set default appearance to false
form.SetDefaultAppearance(false);
//Gets the 'Gender' radio button field
PdfLoadedRadioButtonListField radioButtonField = form.Fields["Gender"] as
PdfLoadedRadioButtonListField;
//Select the item that contains option value as "Male"
```

```

foreach (PdfLoadedRadioButtonItem item in radioButtonField.Items)
{
    //Gets an option value of the item
    if (item.OptionValue == "Male")
    {
        item.Selected = true;
    }
}
//Save the document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
stream.Position = 0;
//Close the document
doc.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
//Get the loaded form
PdfLoadedForm form = doc.Form;
//Set default appearance to false
form.SetDefaultAppearance(false);
//Gets the 'Gender' radio button field
PdfLoadedRadioButtonListField radioButtonField = form.Fields["Gender"] as
PdfLoadedRadioButtonListField;
//Select the item that contains option value as "Male"
foreach (PdfLoadedRadioButtonItem item in radioButtonField.Items)
{
    //Gets an option value of the item
    if (item.OptionValue == "Male")
    {
        item.Selected = true;
    }
}
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)

```

```
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}
```

Adding the list box field

You can create the list box field in PDF forms using [PdfListBoxField](#) class.

Please refer the below code snippet for adding the list box field in new PDF document.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create list box.
PdfListBoxField listBoxField = new PdfListBoxField(page, "list1");
//Set the properties.
listBoxField.Bounds = new RectangleF(100, 60, 100, 50);
//Add the items to the list box.
listBoxField.Items.Add(new PdfListFieldItem("English", "English"));
listBoxField.Items.Add(new PdfListFieldItem("French", "French"));
listBoxField.Items.Add(new PdfListFieldItem("German", "German"));
//Select the item.
listBoxField.SelectedIndex = 2;
//Set the multi select option.
listBoxField.MultiSelect = true;
//Add the list box into PDF document
document.Form.Fields.Add(listBoxField);
//Save the document.
document.Save("Form.pdf");
//close the document
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Add a new page to PDF document.
Dim page As PdfPage = document.Pages.Add()
'Create list box.
Dim listBoxField As New PdfListBoxField(page, "list1")
'Set the properties.
listBoxField.Bounds = New RectangleF(100, 60, 100, 50)
'Add the items to the list box.
listBoxField.Items.Add(New PdfListFieldItem("English", "English"))
listBoxField.Items.Add(New PdfListFieldItem("French", "French"))
listBoxField.Items.Add(New PdfListFieldItem("German", "German"))
'Select the item.
listBoxField.SelectedIndex = 2
```

```
'Set the multi select option.
listBoxField.MultiSelect = True
'Add the list box into PDF document
document.Form.Fields.Add(listBoxField)
'Save the document.
document.Save("Form.pdf")
'close the document
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create list box.
PdfListBoxField listBoxField = new PdfListBoxField(page, "list1");
//Set the properties.
listBoxField.Bounds = new RectangleF(100, 60, 100, 50);
//Add the items to the list box.
listBoxField.Items.Add(new PdfListFieldItem("English", "English"));
listBoxField.Items.Add(new PdfListFieldItem("French", "French"));
listBoxField.Items.Add(new PdfListFieldItem("German", "German"));
//Select the item.
listBoxField.SelectedIndex = 2;
//Set the multi select option.
listBoxField.MultiSelect = true;
//Add the list box into PDF document
document.Form.Fields.Add(listBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Form.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create list box.
PdfListBoxField listBoxField = new PdfListBoxField(page, "list1");
//Set the properties.
listBoxField.Bounds = new Syncfusion.Drawing.RectangleF(100, 60, 100, 50);
//Add the items to the list box.
listBoxField.Items.Add(new PdfListFieldItem("English", "English"));
listBoxField.Items.Add(new PdfListFieldItem("French", "French"));
listBoxField.Items.Add(new PdfListFieldItem("German", "German"));
//Select the item.
listBoxField.SelectedIndex = 0;
//Set the multi select option.
listBoxField.MultiSelect = true;
```

```
//Add the list box into PDF document
document.Form.Fields.Add(listBoxField);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create list box.
PdfListBoxField listBoxField = new PdfListBoxField(page, "list1");
//Set the properties.
listBoxField.Bounds = new Syncfusion.Drawing.RectangleF(100, 60, 100, 50);
//Add the items to the list box.
listBoxField.Items.Add(new PdfListFieldItem("English", "English"));
listBoxField.Items.Add(new PdfListFieldItem("French", "French"));
listBoxField.Items.Add(new PdfListFieldItem("German", "German"));
//Select the item.
listBoxField.SelectedIndex = 2;
//Set the multi select option.
listBoxField.MultiSelect = true;
//Add the list box into PDF document
document.Form.Fields.Add(listBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}
```

Please refer the below code snippet for adding the list box field in existing PDF document.

C#

```
//Load the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create list box.
PdfListBoxField listBoxField = new PdfListBoxField(loadedPage, "list1");
//Set the properties.
listBoxField.Bounds = new RectangleF(100, 60, 100, 50);
//Add the items to the list box.
listBoxField.Items.Add(new PdfListFieldItem("English", "English"));
listBoxField.Items.Add(new PdfListFieldItem("French", "French"));
listBoxField.Items.Add(new PdfListFieldItem("German", "German"));
//Select the item.
listBoxField.SelectedIndex = 2;
//Set the multi select option.
listBoxField.MultiSelect = true;
//Add the list box into PDF document
loadedDocument.Form.Fields.Add(listBoxField);
//Save the document.
loadedDocument.Save("Form.pdf");
//close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Create the form if the form does not exist in the loaded document
If loadedDocument.Form Is Nothing Then
loadedDocument.CreateForm()
End If
'Load the page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Create list box.
Dim listBoxField As New PdfListBoxField(loadedPage, "list1")
'Set the properties.
listBoxField.Bounds = New RectangleF(100, 60, 100, 50)
'Add the items to the list box.
listBoxField.Items.Add(New PdfListFieldItem("English", "English"))
listBoxField.Items.Add(New PdfListFieldItem("French", "French"))
listBoxField.Items.Add(New PdfListFieldItem("German", "German"))
'Select the item.
listBoxField.SelectedIndex = 2
'Set the multi select option.
```



```
listBoxField.MultiSelect = True
'Add the list box into PDF document
loadedDocument.Form.Fields.Add(listBoxField)
'Save the document.
loadedDocument.Save("Form.pdf")
'close the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create list box.
PdfListBoxField listBoxField = new PdfListBoxField(loadedPage, "list1");
//Set the properties.
listBoxField.Bounds = new RectangleF(100, 60, 100, 50);
//Add the items to the list box.
listBoxField.Items.Add(new PdfListFieldItem("English", "English"));
listBoxField.Items.Add(new PdfListFieldItem("French", "French"));
listBoxField.Items.Add(new PdfListFieldItem("German", "German"));
//Select the item.
listBoxField.SelectedIndex = 2;
//Set the multi select option.
listBoxField.MultiSelect = true;
//Add the list box into PDF document
loadedDocument.Form.Fields.Add(listBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Form.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
```

```

loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create list box.
PdfListBoxField listBoxField = new PdfListBoxField(loadedPage, "list1");
//Set the properties.
listBoxField.Bounds = new Syncfusion.Drawing.RectangleF(100, 60, 100, 50);
//Add the items to the list box.
listBoxField.Items.Add(new PdfListFieldItem("English", "English"));
listBoxField.Items.Add(new PdfListFieldItem("French", "French"));
listBoxField.Items.Add(new PdfListFieldItem("German", "German"));
//Select the item.
listBoxField.SelectedIndex = 2;
//Set the multi select option.
listBoxField.MultiSelect = true;
//Add the list box into PDF document
loadedDocument.Form.Fields.Add(listBoxField);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create list box.
PdfListBoxField listBoxField = new PdfListBoxField(loadedPage, "list1");
//Set the properties.
listBoxField.Bounds = new Syncfusion.Drawing.RectangleF(100, 60, 100, 50);
//Add the items to the list box.
listBoxField.Items.Add(new PdfListFieldItem("English", "English"));
listBoxField.Items.Add(new PdfListFieldItem("French", "French"));
listBoxField.Items.Add(new PdfListFieldItem("German", "German"));
//Select the item.
listBoxField.SelectedIndex = 2;
//Set the multi select option.

```

```

listBoxField.MultiSelect = true;
//Add the list box into PDF document
loadedDocument.Form.Fields.Add(listBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}

```

Adding the check Box field

You can create the check box field in PDF forms using [PdfCheckBoxField](#) class.

Please refer the below code snippet for adding the check box field in new PDF document.

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create Check Box field.
PdfCheckBoxField checkBoxField = new PdfCheckBoxField(page, "CheckBox");
//Set check box properties.
checkBoxField.ToolTip = "Check Box";
checkBoxField.Bounds = new RectangleF(0, 20, 10, 10);
//Add the form field to the document.
document.Form.Fields.Add(checkBoxField);
//Save the document.
document.Save("Form.pdf");
//close the document
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
'Add a new page to PDF document.
Dim page As PdfPage = document.Pages.Add()
'Create Check Box field.
Dim checkBoxField As New PdfCheckBoxField(page, "CheckBox")
'Set check box properties.

```

```

checkBoxField.ToolTip = "Check Box"
checkBoxField.Bounds = new RectangleF(0, 20, 10, 10)
'Add the form field to the document.'
document.Form.Fields.Add(checkBoxField)
'Save the document.'
document.Save("Form.pdf")
'close the document'
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create Check Box field.
PdfCheckBoxField checkBoxField = new PdfCheckBoxField(page, "CheckBox");
//Set check box properties.
checkBoxField.ToolTip = "Check Box";
checkBoxField.Bounds = new RectangleF(0, 20, 10, 10);
//Add the form field to the document.
document.Form.Fields.Add(checkBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Form.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create Check Box field.
PdfCheckBoxField checkBoxField = new PdfCheckBoxField(page, "CheckBox");
//Set check box properties.
checkBoxField.ToolTip = "Check Box";
checkBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 20, 10, 10);
//Add the form field to the document.
document.Form.Fields.Add(checkBoxField);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";

```

```
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create Check Box field.
PdfCheckBoxField checkBoxField = new PdfCheckBoxField(page, "CheckBox");
//Set check box properties.
checkBoxField.ToolTip = "Check Box";
checkBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 20, 10, 10);
//Add the form field to the document.
document.Form.Fields.Add(checkBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}
```

Please refer the below code snippet for adding the check box field in existing PDF document.

C#

```
//Load the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Create the form if the form does not exist in the loaded document
if(loadedDocument.Form==null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create Check Box field.
PdfCheckBoxField checkBoxField = new PdfCheckBoxField(loadedPage,
"CheckBox");
//Set check box properties.
checkBoxField.ToolTip = "Check Box";
checkBoxField.Bounds = new RectangleF(0, 20, 10, 10);
//Add the form field to the existing document.
```

```
loadedDocument.Form.Fields.Add(checkBoxField);
//Save the document.
loadedDocument.Save("Form.pdf");
//close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Create the form if the form does not exist in the loaded document
If loadedDocument.Form Is Nothing Then
loadedDocument.CreateForm()
End If
'Load the page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Create Check Box field.
Dim checkBoxField As New PdfCheckBoxField(loadedPage, "CheckBox")
'Set check box properties.
checkBoxField.ToolTip = "Check Box"
checkBoxField.Bounds = New RectangleF(0, 20, 10, 10)
'Add the form field to the existing document.
loadedDocument.Form.Fields.Add(checkBoxField)
'Save the document.
loadedDocument.Save("Form.pdf")
'close the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create Check Box field.
PdfCheckBoxField checkBoxField = new PdfCheckBoxField(loadedPage,
"CheckBox");
//Set check box properties.
checkBoxField.ToolTip = "Check Box";
checkBoxField.Bounds = new RectangleF(0, 20, 10, 10);
//Add the form field to the existing document.
loadedDocument.Form.Fields.Add(checkBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
```

```
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Form.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create Check Box field.
PdfCheckBoxField checkBoxField = new PdfCheckBoxField(loadedPage,
"CheckBox");
//Set check box properties.
checkBoxField.ToolTip = "Check Box";
checkBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 20, 10, 10);
//Add the form field to the existing document.
loadedDocument.Form.Fields.Add(checkBoxField);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create Check Box field.
```

```

PdfCheckBoxField checkBoxField = new PdfCheckBoxField(loadedPage,
"CheckBox");
//Set check box properties.
checkBoxField.ToolTip = "Check Box";
checkBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 20, 10, 10);
//Add the form field to the existing document.
loadedDocument.Form.Fields.Add(checkBoxField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}

```

Adding the signature field

You can add the signature field in PDF forms using [PdfSignatureField](#) class.

Please refer the below code snippet for adding the signature field in new PDF document.

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create PDF Signature field.
PdfSignatureField signatureField = new PdfSignatureField(page, "Signature");
//Set properties to the signature field.
signatureField.Bounds = new RectangleF(0, 400, 90, 20);
signatureField.ToolTip = "Signature";
//Add the form field to the document.
document.Form.Fields.Add(signatureField);
//Save the document.
document.Save("Form.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
'Add a new page to PDF document.

```



```

Dim page As PdfPage = document.Pages.Add()
'Create PDF Signature field.
Dim signatureField As New PdfSignatureField(page, "Signature")
'Set properties to the signature field.
signatureField.Bounds = New RectangleF(0, 400, 90, 20)
signatureField.ToolTip = "Signature"
'Add the form field to the document.
document.Form.Fields.Add(signatureField)
'Save the document.
document.Save("Form.pdf")
'Close the document
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create PDF Signature field.
PdfSignatureField signatureField = new PdfSignatureField(page, "Signature");
//Set properties to the signature field.
signatureField.Bounds = new RectangleF(0, 400, 90, 20);
signatureField.ToolTip = "Signature";
//Add the form field to the document.
document.Form.Fields.Add(signatureField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Form.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create PDF Signature field.
PdfSignatureField signatureField = new PdfSignatureField(page, "Signature");
//Set properties to the signature field.
signatureField.Bounds = new Syncfusion.Drawing.RectangleF(0, 400, 90, 20);
signatureField.ToolTip = "Signature";
//Add the form field to the document.
document.Form.Fields.Add(signatureField);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
document.Close(true);

```

```
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create PDF Signature field.
PdfSignatureField signatureField = new PdfSignatureField(page, "Signature");
//Set properties to the signature field.
signatureField.Bounds = new Syncfusion.Drawing.RectangleF(0, 400, 90, 20);
signatureField.ToolTip = "Signature";
//Add the form field to the document.
document.Form.Fields.Add(signatureField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}
```

Please refer the below code snippet for adding the signature field in existing PDF document.

C#

```
//Load the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Create the form if the form does not exist in the loaded document
if(loadedDocument.Form==null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create PDF Signature field.
PdfSignatureField signatureField = new PdfSignatureField(loadedPage,
"Signature");
```

```
//Set properties to the signature field.
signatureField.Bounds = new RectangleF(0, 400, 90, 20);
signatureField.ToolTip = "Signature";
//Add the form field to the existing document.
loadedDocument.Form.Fields.Add(signatureField);
//Save the document.
loadedDocument.Save("Form.pdf");
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Create the form if the form does not exist in the loaded document
If loadedDocument.Form Is Nothing Then
loadedDocument.CreateForm()
End If
'Load the page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Create PDF Signature field.
Dim signatureField As New PdfSignatureField(loadedPage, "Signature")
'Set properties to the signature field.
signatureField.Bounds = New RectangleF(0, 400, 90, 20)
signatureField.ToolTip = "Signature"
'Add the form field to the existing document.
loadedDocument.Form.Fields.Add(signatureField)
'Save the document.
loadedDocument.Save("Form.pdf")
'Close the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create PDF Signature field.
PdfSignatureField signatureField = new PdfSignatureField(loadedPage,
"Signature");
//Set properties to the signature field.
signatureField.Bounds = new RectangleF(0, 400, 90, 20);
signatureField.ToolTip = "Signature";
```

```
//Add the form field to the existing document.
loadedDocument.Form.Fields.Add(signatureField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Form.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create PDF Signature field.
PdfSignatureField signatureField = new PdfSignatureField(loadedPage,
"Signature");
//Set properties to the signature field.
signatureField.Bounds = new Syncfusion.Drawing.RectangleF(0, 400, 90, 20);
signatureField.ToolTip = "Signature";
//Add the form field to the existing document.
loadedDocument.Form.Fields.Add(signatureField);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
```

```

//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create PDF Signature field.
PdfSignatureField signatureField = new PdfSignatureField(loadedPage,
"Signature");
//Set properties to the signature field.
signatureField.Bounds = new Syncfusion.Drawing.RectangleF(0, 400, 90, 20);
signatureField.ToolTip = "Signature";
//Add the form field to the existing document.
loadedDocument.Form.Fields.Add(signatureField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}

```

Adding the button field

To create button fields in PDF forms, you can use [PdfButtonField](#) class.

The below code illustrates how to add the button field in new PDF document.

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a Button.
PdfButtonField buttonField = new PdfButtonField(page, "Click");
//Set properties to the Button field.
buttonField.Bounds = new RectangleF(0, 150, 90, 20);
buttonField.Text = "Click";
//Add the form field to the document.
document.Form.Fields.Add(buttonField);
//Save the document.
document.Save("Form.pdf");
//close the document
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
'Add a new page to PDF document.
Dim page As PdfPage = document.Pages.Add()
'Create a Button.
Dim buttonField As New PdfButtonField(page, "Click")
'Set properties to the Button field.
buttonField.Bounds = New RectangleF(0, 150, 90, 20)
buttonField.Text = "Click"
'Add the form field to the document.
document.Form.Fields.Add(buttonField)
'Save the document.
document.Save("Form.pdf")
'close the document
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a Button.
PdfButtonField buttonField = new PdfButtonField(page, "Click");
//Set properties to the Button field.
buttonField.Bounds = new RectangleF(0, 150, 90, 20);
buttonField.Text = "Click";
//Add the form field to the document.
document.Form.Fields.Add(buttonField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Form.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a Button.
PdfButtonField buttonField = new PdfButtonField(page, "Click");
//Set properties to the Button field.
buttonField.Bounds = new Syncfusion.Drawing.RectangleF(0, 150, 90, 20);
buttonField.Text = "Click";
//Add the form field to the document.
document.Form.Fields.Add(buttonField);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.

```

```

stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a Button.
PdfButtonField buttonField = new PdfButtonField(page, "Click");
//Set properties to the Button field.
buttonField.Bounds = new Syncfusion.Drawing.RectangleF(0, 150, 90, 20);
buttonField.Text = "Click";
//Add the form field to the document.
document.Form.Fields.Add(buttonField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}

```

Please refer the below code snippet for adding the button field in existing PDF document.

C#

```

//Load the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;

```

```
//Create a Button and set properties to the Button field.
PdfButtonField buttonField = new PdfButtonField(loadedPage, "Click");
buttonField.Bounds = new RectangleF(0, 150, 90, 20);
buttonField.Text = "Click";
//Add the form field to the existing document.
loadedDocument.Form.Fields.Add(buttonField);
//Save the document.
loadedDocument.Save("Form.pdf");
//close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Create the form if the form does not exist in the loaded document
If loadedDocument.Form Is Nothing Then
loadedDocument.CreateForm()
End If
'Load the page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Create a Button and set properties to the Button field.
Dim buttonField As New PdfButtonField(loadedPage, "Click")
buttonField.Bounds = New RectangleF(0, 150, 90, 20)
buttonField.Text = "Click"
'Add the form field to the existing document.
loadedDocument.Form.Fields.Add(buttonField)
'Save the document.
loadedDocument.Save("Form.pdf")
'close the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a Button and set properties to the Button field.
PdfButtonField buttonField = new PdfButtonField(loadedPage, "Click");
buttonField.Bounds = new RectangleF(0, 150, 90, 20);
buttonField.Text = "Click";
//Add the form field to the existing document.
loadedDocument.Form.Fields.Add(buttonField);
```



```
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Form.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a Button and set properties to the Button field.
PdfButtonField buttonField = new PdfButtonField(loadedPage, "Click");
buttonField.Bounds = new Syncfusion.Drawing.RectangleF(0, 150, 90, 20);
buttonField.Text = "Click";
//Add the form field to the existing document.
loadedDocument.Form.Fields.Add(buttonField);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Create the form if the form does not exist in the loaded document
if (loadedDocument.Form == null)
loadedDocument.CreateForm();
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a Button and set properties to the Button field.
PdfButtonField buttonField = new PdfButtonField(loadedPage, "Click");
```

```

buttonField.Bounds = new Syncfusion.Drawing.RectangleF(0, 150, 90, 20);
buttonField.Text = "Click";
//Add the form field to the existing document.
loadedDocument.Form.Fields.Add(buttonField);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}

```

Set appearance to the PDF form fields

After filling the form fields in the PDF document, it may appear empty due to the absence of the appearance dictionary. By setting false to the [SetDefaultAppearance] (<https://help.syncfusion.com/cr/aspnetmvc/Syncfusion.Pdf.Base~Syncfusion.Pdf.Interactive.PdfForm~SetDefaultAppearance.html>) method in PdfForm class, you can create the appearance dictionary. By this, the text will be visible in all PDF Viewers.

The following code snippet explains how to set appearance to the PDF form fields.

C#

```

//Load the PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Set the default appearance
loadedForm.SetDefaultAppearance(false);
//Get the loaded form field
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "Text";
//Save the document
loadedDocument.Save("Form.pdf");
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the PDF document
Dim loadedDocument As New PdfLoadedDocument(fileName)

```

```

'Get the loaded form
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
'Set the default appearance
loadedForm.SetDefaultAppearance(False)
'Get the loaded form field
Dim loadedTextBoxField As PdfLoadedTextBoxField =
TryCast(loadedForm.Fields(0), PdfLoadedTextBoxField)
loadedTextBoxField.Text = "Text"
'Save the document
loadedDocument.Save("Form.pdf")
'Close the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Set the default appearance
loadedForm.SetDefaultAppearance(false);
//Get the loaded form field
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as PdfLoadedTextBoxField;
loadedTextBoxField.Text = "Text"
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the PDF/UWP section for respective code samples
Save(stream, "Form.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open, FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Set the default appearance
loadedForm.SetDefaultAppearance(false);
//Get the loaded form field
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as PdfLoadedTextBoxField;
loadedTextBoxField.Text = "text";

```

```
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the content type for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Set the default appearance
loadedForm.SetDefaultAppearance(false);
//Get the loaded form field
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "text";
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}
```

Modifying the existing form field in PDF document

You can modify an existing form field by getting the field from the [PdfFormFieldCollection](#). You can retrieve a field from the field collection by index or by field name.

The following code snippet explains how to modify an existing form field in a PDF document.

C#

```

//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded form field and modify the properties.
PdfLoadedTextBoxField loadedTextBoxField= loadedForm.Fields[0] as
PdfLoadedTextBoxField;
RectangleF newBounds = new RectangleF(100, 100, 150, 50);
loadedTextBoxField.Bounds = newBounds;
loadedTextBoxField.SpellCheck = true;
loadedTextBoxField.Text = "New text of the field.";
loadedTextBoxField.Password = false;
//Save the document.
loadedDocument.Save("sample.pdf");
//close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Get the loaded form.
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
'Get the loaded form field and Modify the properties.
Dim loadedTextBoxField As PdfLoadedTextBoxField =
TryCast(loadedForm.Fields(0), PdfLoadedTextBoxField)
Dim newBounds As New RectangleF(100, 100, 150, 50)
loadedTextBoxField.Bounds = newBounds
loadedTextBoxField.SpellCheck = True
loadedTextBoxField.Text = "New text of the field."
loadedTextBoxField.Password = False
'Save the document.
loadedDocument.Save("sample.pdf")
'close the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded form field and modify the properties.
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
RectangleF newBounds = new RectangleF(100, 100, 150, 50);

```

```
loadedTextBoxField.Bounds = newBounds;
loadedTextBoxField.SpellCheck = true;
loadedTextBoxField.Text = "New text of the field.";
loadedTextBoxField.Password = false;
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "sample.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded form field and modify the properties.
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
RectangleF newBounds = new RectangleF(100, 100, 150, 50);
loadedTextBoxField.Bounds = newBounds;
loadedTextBoxField.SpellCheck = true;
loadedTextBoxField.Text = "New text of the field.";
loadedTextBoxField.Password = false;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded form field and modify the properties.
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
```

```

Syncfusion.Drawing.RectangleF newBounds = new
Syncfusion.Drawing.RectangleF(100, 100, 150, 50);
loadedTextBoxField.Bounds = newBounds;
loadedTextBoxField.SpellCheck = true;
loadedTextBoxField.Text = "New text of the field.";
loadedTextBoxField.Password = false;
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

Retrieving/Modifying the fore and back color of an existing form fields

You can retrieve/modify the fore and background color of existing form fields in a PDF document by using **ForeColor** and **BackColor** properties of the respective form fields. The following code snippet illustrate this.

C#

```

//Load the PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded form field
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
//Get fore color of the field
PdfColor foreColor = loadedTextBoxField.ForeColor;
//Set the fore color
loadedTextBoxField.ForeColor = new PdfColor(Color.Red);
//Get background color of the field
PdfColor backColor = loadedTextBoxField.BackColor;
//Set the background color
loadedTextBoxField.BackColor = new PdfColor(Color.Green);
//Save the document
loadedDocument.Save("Form.pdf");
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the PDF document
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Get the loaded form
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
'Get the loaded form field
Dim loadedTextBoxField As PdfLoadedTextBoxField =
TryCast(loadedForm.Fields(0), PdfLoadedTextBoxField)
'Get fore color of the field
Dim foreColor As PdfColor = loadedTextBoxField.ForeColor
'Set the fore color
loadedTextBoxField.ForeColor = New PdfColor(Color.Red)
'Get background color of the field
Dim backColor As PdfColor = loadedTextBoxField.BackColor
'Set the background color
loadedTextBoxField.BackColor = New PdfColor(Color.Green)
'Save the document
loadedDocument.Save("Form.pdf")
'Close the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded form field
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
//Get fore color of the field
PdfColor foreColor = loadedTextBoxField.ForeColor;
//Set the fore color
loadedTextBoxField.ForeColor = new PdfColor(255,0,0);
//Get background color of the field
PdfColor backColor = loadedTextBoxField.BackColor;
//Set the background color
loadedTextBoxField.BackColor = new PdfColor(0,255,0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Form.pdf");

```

ASP.NET CORE


```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded form field
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
//Get fore color of the field
PdfColor foreColor = loadedTextBoxField.ForeColor;
//Set the fore color
loadedTextBoxField.ForeColor = new PdfColor(Color.Red);
//Get background color of the field
PdfColor backColor = loadedTextBoxField.BackColor;
//Set the background color
loadedTextBoxField.BackColor = new PdfColor(Color.Green);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the content type for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded form field
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
//Get fore color of the field
PdfColor foreColor = loadedTextBoxField.ForeColor;
//Set the fore color
loadedTextBoxField.ForeColor = new PdfColor(Syncfusion.Drawing.Color.Red);
//Get background color of the field
PdfColor backColor = loadedTextBoxField.BackColor;
//Set the background color
loadedTextBoxField.BackColor = new PdfColor(Syncfusion.Drawing.Color.Green);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close(true);
```

```
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}
```

Filling form fields in an existing PDF Document

Essential PDF allows you to fill the form fields using [PdfLoadedField](#) class.

Filling the text box field

You can fill a text box field using [Text](#) property of [PdfLoadedTextBoxField](#) class. The below code snippet illustrates this.

C#

```
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded text box field and fill it.
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "First Name";
//Save the modified document.
loadedDocument.Save("sample.pdf");
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Get the loaded form.
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
'Get the loaded text box field and fill it.
Dim loadedTextBoxField As PdfLoadedTextBoxField =
TryCast(loadedForm.Fields(0), PdfLoadedTextBoxField)
loadedTextBoxField.Text = "First Name"
'Save the modified document.
loadedDocument.Save("sample.pdf")
'Close the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
```

```

var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded text box field and fill it.
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "First Name";
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "sample.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded text box field and fill it.
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "First Name";
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");

```

```

PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded text box field and fill it.
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "First Name";
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

Filling the combo box field

You can fill a combo box field using [SelectedValue](#) or [SelectedIndex](#) properties of [PdfLoadedComboBoxField](#) class. Please refer the below code snippet to fill the combo box field in an existing PDF document.

C#

```

//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded combo box field and modify the properties
PdfLoadedComboBoxField loadedComboboxField = loadedForm.Fields[1] as
PdfLoadedComboBoxField;
loadedComboboxField.SelectedIndex = 1;
//Save the modified document.
loadedDocument.Save("sample.pdf");
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Get the loaded form.
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
'Get the loaded combo box field and modify the properties

```

```

Dim loadedComboboxField As PdfLoadedComboBoxField =
TryCast(loadedForm.Fields(1), PdfLoadedComboBoxField)
loadedComboboxField.SelectedIndex = 1
'Save the modified document.
loadedDocument.Save("sample.pdf")
'Close the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded combo box field and modify the properties
PdfLoadedComboBoxField loadedComboboxField = loadedForm.Fields[1] as
PdfLoadedComboBoxField;
loadedComboboxField.SelectedIndex = 1;
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "sample.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded combo box field and modify the properties
PdfLoadedComboBoxField loadedComboboxField = loadedForm.Fields[1] as
PdfLoadedComboBoxField;
loadedComboboxField.SelectedIndex = 1;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.

```

```
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded combo box field and modify the properties
PdfLoadedComboBoxField loadedComboboxField = loadedForm.Fields[1] as
PdfLoadedComboBoxField;
loadedComboboxField.SelectedIndex = 1;
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}
```

Filling the radio button field

You can fill a radio button field using [SelectedValue](#) or [SelectedIndex](#) properties of [PdfLoadedRadioButtonListField](#) class. Please refer the below code snippet to fill the radio button field in an existing PDF document.

C#

```
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded radio button field.
PdfLoadedRadioButtonListField loadedRadioButtonField = loadedForm.Fields[3]
as PdfLoadedRadioButtonListField;
loadedRadioButtonField.SelectedIndex = 1;
//Save the modified document.
```

```
loadedDocument.Save("sample.pdf");
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Get the loaded form.
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
'Get the loaded radio button field.
Dim loadedRadioButtonField As PdfLoadedRadioButtonListField =
TryCast(loadedForm.Fields(3), PdfLoadedRadioButtonListField)
loadedRadioButtonField.SelectedIndex = 1
'Save the modified document.
loadedDocument.Save("sample.pdf")
'Close the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded radio button field.
PdfLoadedRadioButtonListField loadedRadioButtonField = loadedForm.Fields[3]
as PdfLoadedRadioButtonListField;
loadedRadioButtonField.SelectedIndex = 1;
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "sample.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded radio button field.
```

```

PdfLoadedRadioButtonListField loadedRadioButtonField = loadedForm.Fields[3]
as PdfLoadedRadioButtonListField;
loadedRadioButtonField.SelectedIndex = 1;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded radio button field.
PdfLoadedRadioButtonListField loadedRadioButtonField = loadedForm.Fields[0]
as PdfLoadedRadioButtonListField;
loadedRadioButtonField.SelectedIndex = 1;
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

Filling the list box field

The below code snippet illustrates how to fill the list box field in an existing PDF document, using [SelectedIndex](#) property of [PdfLoadedListBoxField](#) class.

C#

```

//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Fill list box.
PdfLoadedListBoxField loadedListBox = loadedForm.Fields[2] as
PdfLoadedListBoxField;
// Fill list box and Modify the list box select index
loadedListBox.SelectedIndex = new int[2] { 1, 2 };
//Save the modified document.
loadedDocument.Save("sample.pdf");
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Get the loaded form.
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
'Fill list box and Modify the list box select index
Dim loadedListBox As PdfLoadedListBoxField = TryCast(loadedForm.Fields(0),
PdfLoadedListBoxField)
loadedListBox.SelectedIndex = New Integer(1) {1, 2}
'Save the modified document.
loadedDocument.Save("sample.pdf")
'close the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Fill list box.
PdfLoadedListBoxField loadedListBox = loadedForm.Fields[2] as
PdfLoadedListBoxField;
// Fill list box and Modify the list box select index
loadedListBox.SelectedIndex = new int[2] { 1, 2 };
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);

```

```
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "sample.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Fill list box.
PdfLoadedListBoxField loadedListBox = loadedForm.Fields[2] as
PdfLoadedListBoxField;
// Fill list box and Modify the list box select index
loadedListBox.SelectedIndex = new int[2] { 1, 2 };
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Fill list box.
PdfLoadedListBoxField loadedListBox = loadedForm.Fields[2] as
PdfLoadedListBoxField;
// Fill list box and Modify the list box select index
loadedListBox.SelectedIndex = new int[2] { 1, 2 };
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
```

```
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}
```

Filling the check Box field

You can fill a check box field by enabling [Checked](#) property of [PdfLoadedCheckBoxField](#) class. Please refer the below code snippet to fill the check box field.

C#

```
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//load the check box from field collection
PdfLoadedCheckBoxField loadedCheckBoxField = loadedForm.Fields[0] as
PdfLoadedCheckBoxField;
// fill the checkbox .
loadedCheckBoxField.Items[0].Checked = true;
//Check the checkbox if it is not grouped.
loadedCheckBoxField.Checked = true;
//Save the modified document.
loadedDocument.Save("sample.pdf");
//close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Get the loaded form.
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
'load the check box from field collection
Dim loadedCheckBoxField As PdfLoadedCheckBoxField =
TryCast(loadedForm.Fields(0), PdfLoadedCheckBoxField)
'Fill the checkbox.
loadedCheckBoxField.Items(0).Checked = True
'Check the checkbox if it is not grouped.
loadedCheckBoxField.Checked = True
'Save the modified document.
loadedDocument.Save("sample.pdf")
'Close the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
```

```

//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//load the check box from field collection
PdfLoadedCheckBoxField loadedCheckBoxField = loadedForm.Fields[0] as PdfLoadedCheckBoxField;
// fill the checkbox .
loadedCheckBoxField.Items[0].Checked = true;
//Check the checkbox if it is not grouped.
loadedCheckBoxField.Checked = true;
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples.
Save(stream, "sample.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open, FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//load the check box from field collection
PdfLoadedCheckBoxField loadedCheckBoxField = loadedForm.Fields[0] as PdfLoadedCheckBoxField;
// fill the checkbox .
loadedCheckBoxField.Items[0].Checked = true;
//Check the checkbox if it is not grouped.
loadedCheckBoxField.Checked = true;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//load the check box from field collection
PdfLoadedCheckBoxField loadedCheckBoxField = loadedForm.Fields[0] as
PdfLoadedCheckBoxField;
// fill the checkbox .
loadedCheckBoxField.Items[0].Checked = true;
//Check the checkbox if it is not grouped.
loadedCheckBoxField.Checked = true;
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

Filling the signature field

The below code snippet illustrates how to fill the signature field with certificate using [Certificate](#) property.

C#

```

//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Create PDF Certificate.
PdfCertificate certificate = new PdfCertificate("Pdf.pfx", "password");
//Load the signature field from field collection and fill this with
certificate
PdfLoadedSignatureField loadedSignatureField = loadedForm.Fields[0] as
PdfLoadedSignatureField;
loadedSignatureField.Signature = new PdfSignature(loadedPage);
loadedSignatureField.Signature.Certificate = certificate;
loadedSignatureField.Signature.Reason = "Reason";
//Save the modified document.

```

```
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Get the loaded form.
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
'Create PDF Certificate.
Dim certificate As New PdfCertificate("Pdf.pfx", "password")
'Load the signature field from field collection and fill with certificate
Dim loadedSignatureField As PdfLoadedSignatureField =
TryCast(loadedForm.Fields(0), PdfLoadedSignatureField)
loadedSignatureField.Signature = New PdfSignature(loadedPage)
loadedSignatureField.Signature.Certificate = certificate
loadedSignatureField.Signature.Reason = "Reason"
'Save the modified document.
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Create PDF Certificate
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.PDF.pfx");
PdfCertificate certificate = new PdfCertificate(certificateStream,
"syncfusion");
//Load the signature field from field collection and fill this with
certificate
PdfLoadedSignatureField loadedSignatureField = loadedForm.Fields[9] as
PdfLoadedSignatureField;
loadedSignatureField.Signature = new PdfSignature();
loadedSignatureField.Signature.Certificate = certificate;
loadedSignatureField.Signature.Reason = "Reason";
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
```

```
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Create PDF Certificate
FileStream certificateStream = new FileStream("PDF.pfx", FileMode.Open,
FileAccess.Read);
PdfCertificate certificate = new PdfCertificate(certificateStream,
"syncfusion");
//Load the signature field from field collection and fill this with
certificate
PdfLoadedSignatureField loadedSignatureField = loadedForm.Fields[9] as
PdfLoadedSignatureField
loadedSignatureField.Signature = new PdfSignature();
loadedSignatureField.Signature.Certificate = certificate;
loadedSignatureField.Signature.Reason = "Reason";
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Create PDF Certificate
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.PDF.pfx");
```

```

PdfCertificate certificate = new PdfCertificate(certificateStream,
"syncfusion");
//Load the signature field from field collection and fill this with
certificate
PdfLoadedSignatureField loadedSignatureField = loadedForm.Fields[9] as
PdfLoadedSignatureField;
loadedSignatureField.Signature = new PdfSignature();
loadedSignatureField.Signature.Certificate = certificate;
loadedSignatureField.Signature.Reason = "Reason";
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
"application/pdf", stream);
}

```

Enumerate the form fields

All the form fields are maintained in [PdfLoadedFormFieldCollection](#) class. You can enumerate the fields from this form field collection and fill them.

The following code example illustrates this.

C#

```

//Load the PDF document.
PdfLoadedDocument document = new PdfLoadedDocument(fileName);
//Get the loaded form.
PdfLoadedForm form = document.Form;
PdfLoadedFormFieldCollection fields = form.Fields;
//Enumerates the form fields.
for (int i = 0; i < fields.Count; i++)
{
if (fields[i] is PdfLoadedTextBoxField)
{
PdfLoadedTextBoxField loadedTextBoxField = fields[i] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "Text";
}
}
//Save and close the modified document.
document.Save(OutputFileName);
document.Close(true);

```


VB.NET

```

'Load the PDF document.
Dim document As New PdfLoadedDocument(fileName)
'Get the loaded form.
Dim form As PdfLoadedForm = document.Form
Dim fields As PdfLoadedFormFieldCollection = form.Fields
'Enumerate the form fields.
For i As Integer = 0 To fields.Count - 1
If TypeOf fields(i) Is PdfLoadedTextBoxField Then
Dim loadedTextBoxField As PdfLoadedTextBoxField = TryCast(fields(i),
PdfLoadedTextBoxField)
loadedTextBoxField.Text = "Text"
End If
Next i
'Save and close the modified document.
document.Save(OutputFileName)
document.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await document.OpenAsync(file);
//Get the loaded form.
PdfLoadedForm form = document.Form;
PdfLoadedFormFieldCollection fields = form.Fields;
//Enumerates the form fields.
for (int i = 0; i < fields.Count; i++)
{
if (fields[i] is PdfLoadedTextBoxField)
{
PdfLoadedTextBoxField loadedTextBoxField = fields[i] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "Text";
}
}
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "output.pdf");

```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm form = document.Form;
PdfLoadedFormFieldCollection fields = form.Fields;
//Enumerates the form fields.
for (int i = 0; i < fields.Count; i++)
{
    if (fields[i] is PdfLoadedTextBoxField)
    {
        PdfLoadedTextBoxField loadedTextBoxField = fields[i] as
        PdfLoadedTextBoxField;
        loadedTextBoxField.Text = "Text";
    }
}
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm form = document.Form;
PdfLoadedFormFieldCollection fields = form.Fields;
//Enumerates the form fields.
for (int i = 0; i < fields.Count; i++)
{
    if (fields[i] is PdfLoadedTextBoxField)
    {
        PdfLoadedTextBoxField loadedTextBoxField = fields[i] as
        PdfLoadedTextBoxField;
        loadedTextBoxField.Text = "Text";
    }
}
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
```

```
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
"application/pdf", stream);
}
```

TryGetField

To get a form field from an existing document using the field name, you can use the [TryGetField](#) method in the [PdfFormFieldCollection](#) class. It specifies whether the particular field is available in the form or not by returning a boolean value.

The below code snippet explains how to get the field from collection using [TryGetField](#) method.

C#

```
// Load the document.
PdfLoadedDocument doc = new PdfLoadedDocument(fileName);
// Load the form from the loaded document.
PdfLoadedForm form = doc.Form;
// Load the form field collections from the form.
PdfLoadedFormFieldCollection fieldCollection = form.Fields as
PdfLoadedFormFieldCollection;
PdfLoadedField loadedField = null;
// Get the field using TryGetField Method.
if (fieldCollection.TryGetField("f1-1", out loadedField))
{
(loadedField as PdfLoadedTextBoxField).Text = "1";
}
//Save and close the modified document.
doc.Save("output.pdf");
doc.Close(true);
```

VB.NET

```
' Load the document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
' Load the form from the loaded document.
Dim form As PdfLoadedForm = loadedDocument.Form
' Load the form field collections from the form.
Dim fieldCollection As PdfLoadedFormFieldCollection = TryCast(form.Fields,
PdfLoadedFormFieldCollection)
Dim loadedField As PdfLoadedField = Nothing
' Get the field using TryGetField Method.
If fieldCollection.TryGetField("f1-1", loadedField) Then
TryCast(loadedField, PdfLoadedTextBoxField).Text = "1"
```

End If

```
'Save and close the modified document.
loadedDocument.Save("output.pdf")
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument doc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await doc.OpenAsync(file);
// Load the form from the loaded document.
PdfLoadedForm form = doc.Form;
// Load the form field collections from the form.
PdfLoadedFormFieldCollection fieldCollection = form.Fields as
PdfLoadedFormFieldCollection;
PdfLoadedField loadedField = null;
// Get the field using TryGetField Method.
if (fieldCollection.TryGetField("f1-1", out loadedField))
{
    (loadedField as PdfLoadedTextBoxField).Text = "1";
}
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document.
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
// Load the form from the loaded document.
PdfLoadedForm form = doc.Form;
// Load the form field collections from the form.
PdfLoadedFormFieldCollection fieldCollection = form.Fields as
PdfLoadedFormFieldCollection;
PdfLoadedField loadedField = null;
// Get the field using TryGetField Method.
if (fieldCollection.TryGetField("f1-1", out loadedField))
{
    (loadedField as PdfLoadedTextBoxField).Text = "1";
}
//Save the document into stream.
MemoryStream stream = new MemoryStream();
```

```

doc.Save(stream);
stream.Position = 0;
//Close the document.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument doc = new PdfLoadedDocument(docStream);
// Load the form from the loaded document.
PdfLoadedForm form = doc.Form;
// Load the form field collections from the form.
PdfLoadedFormFieldCollection fieldCollection = form.Fields as
PdfLoadedFormFieldCollection;
PdfLoadedField loadedField = null;
// Get the field using TryGetField Method.
if (fieldCollection.TryGetField("f1-1", out loadedField))
{
    (loadedField as PdfLoadedTextBoxField).Text = "1";
}
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document.
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
"application/pdf", stream);
}

```

TryGetValue

To get the field value from the given field name, you can use [TryGetValue](#) method in [PdfFormFieldCollection](#) class. It specifies whether the particular field is available in the form or not by returning a boolean value.

Please refer the below code snippet to get the field value from collection using `TryGetValue` method.

C#

```
// Load the document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(filename);
// Load the form from the loaded document.
PdfLoadedForm form = loadedDocument.Form;
// Load the form field collections from the form.
PdfLoadedFormFieldCollection fieldCollection = form.Fields as
PdfLoadedFormFieldCollection;
string fieldValue = string.Empty;
// Get the field value using TryGetValue Method
fieldCollection.TryGetValue("f1-2", out fieldValue);
//Save and close the modified document.
loadedDocument.Save("output.pdf");
loadedDocument.Close(true);
```

VB.NET

```
' Load the document.
Dim loadedDocument As New PdfLoadedDocument(filename)
' Load the form from the loaded document.
Dim form As PdfLoadedForm = loadedDocument.Form
' Load the form field collections from the form.
Dim fieldCollection As PdfLoadedFormFieldCollection = TryCast(form.Fields,
PdfLoadedFormFieldCollection)
Dim fieldValue As String = String.Empty
' Get the field value using TryGetValue Method
fieldCollection.TryGetValue("f1-2", fieldValue)
'Save and close the modified document.
loadedDocument.Save("output.pdf")
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
// Load the form from the loaded document
PdfLoadedForm form = loadedDocument.Form;
// Load the form field collections from the form
PdfLoadedFormFieldCollection fieldCollection = form.Fields as
PdfLoadedFormFieldCollection;
string fieldValue = string.Empty;
// Get the field value using TryGetValue Method
fieldCollection.TryGetValue("f1-2", out fieldValue);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
```

```
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
// Load the form from the loaded document.
PdfLoadedForm form = loadedDocument.Form;
// Load the form field collections from the form.
PdfLoadedFormFieldCollection fieldCollection = form.Fields as
PdfLoadedFormFieldCollection;
string fieldValue = string.Empty;
// Get the field value using TryGetValue Method
fieldCollection.TryGetValue("FirstName", out fieldValue);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
// Load the form from the loaded document.
PdfLoadedForm form = loadedDocument.Form;
// Load the form field collections from the form.
PdfLoadedFormFieldCollection fieldCollection = form.Fields as
PdfLoadedFormFieldCollection;
string fieldValue = string.Empty;
// Get the field value using TryGetValue Method
fieldCollection.TryGetValue("FirstName", out fieldValue);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android
//and iOS platforms. Please refer PDF/Xamarin section for respective code
//samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
"application/pdf", stream);
}
```

Fill the XFA form fields along with Acroform in a same API

The static XFA document contains both the XFA and Acroform.

The Essential PDF supports filling both the XFA and Acroform in a same instance (Fills the XFA form via Acroform instance) by enabling the [EnableXfaFormFill](#) property available in the [PdfLoadedForm](#) class.

The following code snippet illustrates how to fill XFA forms via Acroform API

C#

```
//Load the existing XFA PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Form.pdf");
//Get the existing Acroform
PdfLoadedForm acroform = loadedDocument.Form;
//Enable XFA form filling
acroform.EnableXfaFormFill = true;
//Get the existing text box field
PdfLoadedTextBoxField firstName = acroform.Fields["FirstName"] as
PdfLoadedTextBoxField;
//Set text
firstName.Text = "Simon";
PdfLoadedTextBoxField lastName = acroform.Fields["LastName"] as
PdfLoadedTextBoxField;
//Set text
lastName.Text = "Bistro";
//Save the document
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing XFA PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Form.pdf")
'Get the existing Acroform
Dim acroform As PdfLoadedForm = loadedDocument.Form
'Enable XFA form filling
acroform.EnableXfaFormFill = True
'Get the existing text box field
Dim firstName As PdfLoadedTextBoxField =
TryCast(acroform.Fields("FirstName"), PdfLoadedTextBoxField)
```



```
'Set text
firstName.Text = "Simon"
Dim lastName As PdfLoadedTextBoxField = TryCast(acroform.Fields("LastName"),
PdfLoadedTextBoxField)
'Set text
lastName.Text = "Bistro"
'Save the document
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Get the existing Acroform
PdfLoadedForm acroform = loadedDocument.Form;
//Enable XFA form filling
acroform.EnableXfaFormFill = true;
//Get the existing text box field
PdfLoadedTextBoxField firstName = acroform.Fields["FirstName"] as
PdfLoadedTextBoxField;
//Set text
firstName.Text = "Simon";
PdfLoadedTextBoxField lastName = acroform.Fields["LastName"] as
PdfLoadedTextBoxField;
//Set text
lastName.Text = "Bistro";
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Form.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the existing Acroform
PdfLoadedForm acroform = loadedDocument.Form;
//Enable XFA form filling
acroform.EnableXfaFormFill = true;
//Get the existing text box field
PdfLoadedTextBoxField firstName = acroform.Fields["FirstName"] as
PdfLoadedTextBoxField;
//Set text
```

```

firstName.Text = "Simon";
PdfLoadedTextBoxField lastName = acroform.Fields["LastName"] as
PdfLoadedTextBoxField;
//Set text
lastName.Text = "Bistro";
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
loadedDocument.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

Removing editing capability of form fields

The form field editing or filling capabilities can be removed by either flattening the PDF document or by marking the form or field as read only.

Essential PDF provides support to [Flatten](#) a form field by removing the existing form field and replacing it with graphical objects that would resemble the form field and cannot be edited.

Please refer the sample for flattening the form fields in new PDF document.

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a Text box field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Flatten the whole form field
document.Form.Flatten = true;
//Flattens the first field //form.Fields[0].Flatten = true;
//Add the form field to the document.
document.Form.Fields.Add(textBoxField);
//Save the document.
document.Save("Form.pdf");
//close the document
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
'Add a new page to PDF document.
Dim page As PdfPage = document.Pages.Add()

```

```

'Create a Text box field and add the properties.
Dim textBoxField As New PdfTextBoxField(page, "FirstName")
textBoxField.Bounds = New RectangleF(0, 0, 100, 20)
textBoxField.ToolTip = "First Name"
'Flatten the whole form field
document.Form.Flatten = True
'Add the form field to the document.
document.Form.Fields.Add(textBoxField)
'Save the PDF document.
document.Save("Form.pdf")
'close the document
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new page to PDF document
PdfPage page = document.Pages.Add();
//Create a Text box field and add the properties
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Flatten the whole form field
document.Form.Flatten = true;
//Flattens the first field //form.Fields[0].Flatten = true;
//Add the form field to the document
document.Form.Fields.Add(textBoxField);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Form.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new page to PDF document
PdfPage page = document.Pages.Add();
//Create a Text box field and add the properties
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Flatten the whole form field
document.Form.Flatten = true;
//Flattens the first field //form.Fields[0].Flatten = true;
//Add the form field to the document
document.Form.Fields.Add(textBoxField);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;

```

```
//Close the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name.
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new page to PDF document
PdfPage page = document.Pages.Add();
//Create a Text box field and add the properties
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Flatten the whole form field
document.Form.Flatten = true;
//Flattens the first field //form.Fields[0].Flatten = true;
//Add the form field to the document
document.Form.Fields.Add(textBoxField);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
sample
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}
```

Please refer the sample for flattening the form fields in existing PDF document.

C#

```
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
PdfLoadedFormFieldCollection fields = loadedForm.Fields;
//Get the loaded text box field
```

```

PdfLoadedTextBoxField loadedTextBoxField = fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "Text";
//Flatten the whole form.
loadedForm.Flatten = true;
//Save and close the modified document.
loadedDocument.Save("output.pdf");
loadedDocument.Close(true);

```

VB.NET

```

'Load the PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Get the loaded form.
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
Dim fields As PdfLoadedFormFieldCollection = loadedForm.Fields
'Get the loaded text box field
Dim loadedTextBoxField As PdfLoadedTextBoxField = TryCast(fields(0),
PdfLoadedTextBoxField)
loadedTextBoxField.Text = "Text"
'Flatten the whole form.
loadedForm.Flatten = True
'Save and close the modified document.
loadedDocument.Save("output.pdf")
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
PdfLoadedFormFieldCollection fields = loadedForm.Fields;
//Get the loaded text box field
PdfLoadedTextBoxField loadedTextBoxField = fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "Text";
//Flatten the whole form
loadedForm.Flatten = true;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
PdfLoadedFormFieldCollection fields = loadedForm.Fields;
//Get the loaded text box field
PdfLoadedTextBoxField loadedTextBoxField = fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "Text";
//Flatten the whole form.
loadedForm.Flatten = true;
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
PdfLoadedFormFieldCollection fields = loadedForm.Fields;
//Get the loaded text box field
PdfLoadedTextBoxField loadedTextBoxField = fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "Text";
//Flatten the whole form.
loadedForm.Flatten = true;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
"application/pdf", stream);
}

```

To prevent the user from changing the form field content, you can also use [ReadOnly](#) property.

The below code snippet illustrates how to set the ReadOnly property to a new PDF document.

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create the form
PdfForm form = document.Form;
//Set the form as read only
form.ReadOnly = true;
//Create a text box field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
textBoxField.Text = "john";
//set read only property for a particular field
//textBoxField.ReadOnly = true;
//Add the form field to the document.
document.Form.Fields.Add(textBoxField);
//Save the document.
document.Save("Form.pdf");
//close the document
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
'Add a new page to PDF document.
Dim page As PdfPage = document.Pages.Add()
'Create the form
Dim form As PdfForm = document.Form
'Set the form as read only
form.ReadOnly = True
'Create a Text box field and add the properties.
Dim textBoxField As New PdfTextBoxField(page, "FirstName")
textBoxField.Bounds = New RectangleF(0, 0, 100, 20)
textBoxField.ToolTip = "First Name"
textBoxField.Text = "john"
'set read only property for a particular field

```

```
'textBoxField.ReadOnly = True;
'Add the form field to the document.
document.Form.Fields.Add(textBoxField)
'Save the document.
document.Save("Form.pdf")
'close the document
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create the form
PdfForm form = document.Form;
//Set the form as read only
form.ReadOnly = true;
//Create a text box field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
textBoxField.Text = "john";
//set read only property for a particular field
//textBoxField.ReadOnly = true;
//Add the form field to the document.
document.Form.Fields.Add(textBoxField);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Form.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create the form
PdfForm form = document.Form;
//Set the form as read only
form.ReadOnly = true;
//Create a text box field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
textBoxField.Text = "john";
//set read only property for a particular field
//textBoxField.ReadOnly = true;
//Add the form field to the document.
document.Form.Fields.Add(textBoxField);
//Save the document into stream
```



```

MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create the form
PdfForm form = document.Form;
//Set the form as read only
form.ReadOnly = true;
//Create a text box field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
textBoxField.Text = "john";
//set read only property for a particular field
//textBoxField.ReadOnly = true;
//Add the form field to the document.
document.Form.Fields.Add(textBoxField);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}

```

The below code snippet illustrates how to set the ReadOnly property to an existing PDF document.

C#

```
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Set the form as read only
loadedForm.ReadOnly = true;
//Save the document.
loadedDocument.Save("sample.pdf");
//close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Get the loaded form.
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
'Set the form as read only
loadedForm.ReadOnly = True
'Save the document.
loadedDocument.Save("Form1.pdf")
'close the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Set the form as read only
loadedForm.ReadOnly = true;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "sample.pdf");
```

ASP.NET CORE

```
//Load the PDF document
```

```

FileStream docStream = new FileStream(fileName, FileMode.Open,
    FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Set the form as read only
loadedForm.ReadOnly = true;
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Set the form as read only
loadedForm.ReadOnly = true;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
        "application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
        "application/pdf", stream);
}

```

Note: Flattening and adding Read-only properties can be done to the entire form or an individual form field.

Removing the form fields from existing PDF document

You can remove the form fields from an existing PDF document using [Remove](#) or [RemoveAt](#) methods of [PdfFieldCollection](#) class.

The below code illustrates how to remove the form fields from the existing PDF document.

C#

```
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Load the textbox field
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
//Remove the field
loadedForm.Fields.Remove(loadedTextBoxField);
//Remove the field at index 0
loadedForm.Fields.RemoveAt(0);
//Save the modified document.
loadedDocument.Save("form.pdf");
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Load the page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Get the loaded form.
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
'Load the textbox field
Dim loadedTextBoxField As PdfLoadedTextBoxField =
TryCast(loadedForm.Fields(0), PdfLoadedTextBoxField)
'Remove the field
loadedForm.Fields.Remove(loadedTextBoxField)
'Remove the field at index 0
loadedForm.Fields.RemoveAt(0);
'Save the modified document.
loadedDocument.Save("form.pdf")
'Close the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
```

```

//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Load the textbox field
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as PdfLoadedTextBoxField;
//Remove the field
loadedForm.Fields.Remove(loadedTextBoxField);
//Remove the field at index 0
loadedForm.Fields.RemoveAt(0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code sample
Save(stream, "Form.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Load the textbox field
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as PdfLoadedTextBoxField;
//Remove the field
loadedForm.Fields.Remove(loadedTextBoxField);
//Remove the field at index 0
loadedForm.Fields.RemoveAt(0);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Load the textbox field
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
//Remove the field
loadedForm.Fields.Remove(loadedTextBoxField);
//Remove the field at index 0
loadedForm.Fields.RemoveAt(0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}

```

Importing FDF file to PDF

FDF stands for Forms Data Format. FDF is a file format for representing form data and annotations that are contained in a PDF form. You can import the FDF file to PDF using [ImportDataFDF](#) method in [PdfLoadedForm](#) class.

The below code illustrates how to import FDF file to PDF.

C#

```

//Load an existing document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
// Load the existing form
PdfLoadedForm loadedForm = loadedDocument.Form;
// Load the FDF file
FileStream stream = new FileStream("ImportFDF.fdf", FileMode.Open);
// Import the FDF stream
loadedForm.ImportDataFDF(stream, true);
//Close the document.
loadedDocument.Close(true);

```

VB.NET

```
'Load an existing document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
' Load the existing form
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
' Load the FDF file
Dim stream As New FileStream("ImportFDF.fdf", FileMode.Open)
' Import the FDF stream
loadedForm.ImportDataFDF(stream, True)
' Close the document
loadedDocument.Close(True)
```

Export PDF file to FDF

To export the FDF file from PDF document, you can use [ExportData](#) method in [PdfLoadedForm](#) class.

The below code illustrates how to export FDF file from PDF document.

C#

```
// Load an existing document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
// Load an existing form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Export the existing PDF document to FDF file
loadedForm.ExportData("Export.fdf", DataFormat.Fdf, "SourceForm.pdf");
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
' Load an existing document
Dim loadedDocument As New PdfLoadedDocument(fileName)
' Load an existing form
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
'Export the existing PDF document to FDF file
loadedForm.ExportData("Export.fdf", DataFormat.Fdf, "SourceForm.pdf")
'Close the document
loadedDocument.Close(True)
```

Complex script support for form fields

You can add a complex script language text in PDF AcroForm fields by using the [ComplexScript](#) property of the form field instance. The following code snippet illustrates this.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new PDF page
PdfPage page = document.Pages.Add();
//Create new PDF text box field
PdfTextBoxField textField = new PdfTextBoxField(page, "textBox");
//Set bounds
textField.Bounds = new RectangleF(10, 10, 200, 30);
```

```
//Set text
textField.Text = "สวัสดีชาวโลก";
//Create new PdfTrueTypeFont instance
PdfTrueTypeFont font = new PdfTrueTypeFont(new Font("Tahoma", 10), true);
//Set font
textField.Font = font;
//Enable complex script layout
textField.ComplexScript = true;
//Add the text box field to the form collection
document.Form.Fields.Add(textField);
//Set default appearance as false
document.Form.SetDefaultAppearance(false);
//Save the PDF document
document.Save("Form.pdf");
//Close the document
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As New PdfDocument()
'Add a new PDF page
Dim page As PdfPage = document.Pages.Add()
'Create new PDF text box field
Dim textField As New PdfTextBoxField(page, "textBox")
'Set bounds
textField.Bounds = New RectangleF(10, 10, 200, 30)
'Set text
textField.Text = "สวัสดีชาวโลก"
'Create new PdfTrueTypeFont instance
Dim font As New PdfTrueTypeFont(New Font("Tahoma", 10), True)
'Set font
textField.Font = font
'Enable complex script layout
textField.ComplexScript = True
'Add the text box field to the form collection
document.Form.Fields.Add(textField)
'Set default appearance as false
document.Form.SetDefaultAppearance(False)
'Save the PDF document
document.Save("Form.pdf")
'Close the document
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new PDF page
PdfPage page = document.Pages.Add();
//Create new PDF text box field
PdfTextBoxField textField = new PdfTextBoxField(page, "textBox");
//Set bounds
textField.Bounds = new RectangleF(10, 10, 200, 30);
//Set text
```



```

textField.Text = "สวัสดิ์ชาวโลก";
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("ComplexSc
riptSample.Assets.tahoma.ttf");
//Create a new PDF font instance
PdfFont font = new PdfTrueTypeFont(fontStream, 10);
//Set font
textField.Font = font;
//Enable complex script layout
textField.ComplexScript = true;
//Add the text box field to the form collection
document.Form.Fields.Add(textField);
//Set default appearance to false
document.Form.SetDefaultAppearance(false);
//Save the PDF document
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the PDF document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new PDF page
PdfPage page = document.Pages.Add();
//Create the new PDF text box field
PdfTextBoxField textField = new PdfTextBoxField(page, "textBox");
//Set bounds
textField.Bounds = new RectangleF(10, 10, 200, 30);
//Set text
textField.Text = "สวัสดิ์ชาวโลก";
FileStream fontStream = new FileStream("tahoma.ttf", FileMode.Open,
FileAccess.Read);
//Create a new PDF font instance
PdfFont font = new PdfTrueTypeFont(fontStream, 10);
//Set font
textField.Font = font;
//Enable complex script layout
textField.ComplexScript = true;
//Add the text box field to the form collection
document.Form.Fields.Add(textField);
//Set default appearance as false
document.Form.SetDefaultAppearance(false);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the PDF document
document.Close(true);
//Defining the content type for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";

```

```
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new PDF page
PdfPage page = document.Pages.Add();
//Create the new PDF text box field
PdfTextBoxField textField = new PdfTextBoxField(page, "textBox");
//Set bounds
textField.Bounds = new RectangleF(10, 10, 200, 30);
//Set text
textField.Text = "สวัสดิ์ชาวโลก";
//Load the font as stream
Stream fontStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
tahoma.ttf");
//Create a new PDF font instance
PdfFont font = new PdfTrueTypeFont(fontStream, 10);
//Set font
textField.Font = font;
//Enable complex script layout
textField.ComplexScript = true;
//Add the text box field to the form collection
document.Form.Fields.Add(textField);
//Set default appearance as false
document.Form.SetDefaultAppearance(false);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the PDF document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

You can add the complex script for all the supported form fields by enabling the [ComplexScript](#) property of [PdfForm](#) or [PdfLoadedForm](#) instance.

Supported form fields:

- Text box field
- Combo box field
- List box field
- Button field

The following code example illustrates how to add complex script support for all the supported fields in PDF document.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new PDF page
PdfPage page = document.Pages.Add();
//Create the new PDF text box field
PdfTextBoxField textField = new PdfTextBoxField(page, "textBox");
//Set bounds
textField.Bounds = new RectangleF(10, 10, 200, 30);
//Set text
textField.Text = "สวัสดีชาวโลก";
//Create new PdfTrueTypeFont instance
PdfTrueTypeFont font = new PdfTrueTypeFont(new Font("Tahoma", 10), true);
//Set font
textField.Font = font;
//Add the text box field to the form collection
document.Form.Fields.Add(textField);
//Set default appearance to false
document.Form.SetDefaultAppearance(false);
//Enable complex script layout for form
document.Form.ComplexScript = true;
//Save the PDF document
document.Save("Form.pdf");
//Close the document
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As New PdfDocument()
'Add a new PDF page
Dim page As PdfPage = document.Pages.Add()
'Create the new PDF text box field
Dim textField As New PdfTextBoxField(page, "textBox")
'Set bounds
textField.Bounds = New RectangleF(10, 10, 200, 30)
'Set text
textField.Text = "สวัสดีชาวโลก"
'Create new PdfTrueTypeFont instance
Dim font As New PdfTrueTypeFont(New Font("Tahoma", 10), True)
'Set font
textField.Font = font
'Add the text box field to the form collection
document.Form.Fields.Add(textField)
'Set default appearance to false
document.Form.SetDefaultAppearance(False)
```

```
'Enable complex script layout for form
document.Form.ComplexScript = True
'Save the PDF document
document.Save("Form.pdf")
'Close the document
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new PDF page
PdfPage page = document.Pages.Add();
//Create the new PDF text box field
PdfTextBoxField textField = new PdfTextBoxField(page, "textBox");
//Set bounds
textField.Bounds = new RectangleF(10, 10, 200, 30);
//Set text
textField.Text = "สวัสดิ์ชาวโลก";
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("ComplexSc
riptSample.Assets.tahoma.ttf");
//Create a new PDF font instance
PdfFont font = new PdfTrueTypeFont(fontStream, 10);
//Set font
textField.Font = font;
//Add the text box field to the form collection
document.Form.Fields.Add(textField);
//Set default appearance to false
document.Form.SetDefaultAppearance(false);
//Enable complex script layout for form
document.Form.ComplexScript = true;
//Save the PDF document
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new PDF page
PdfPage page = document.Pages.Add();
//Create the new PDF text box field
PdfTextBoxField textField = new PdfTextBoxField(page, "textBox");
//Set bounds
textField.Bounds = new RectangleF(10, 10, 200, 30);
//Set text
textField.Text = "สวัสดิ์ชาวโลก";
FileStream fontStream = new FileStream("tahoma.ttf", FileMode.Open,
FileAccess.Read);
```

```
//Create a new PDF font instance
PdfFont font = new PdfTrueTypeFont(fontStream, 10);
//Set font
textField.Font = font;
//Add the text box field to the form collection
document.Form.Fields.Add(textField);
//Set default appearance as false
document.Form.SetDefaultAppearance(false);
//Enable complex script layout for form
document.Form.ComplexScript = true;
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the PDF document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new PDF page
PdfPage page = document.Pages.Add();
//Create the new PDF text box field
PdfTextBoxField textField = new PdfTextBoxField(page, "textBox");
//Set bounds
textField.Bounds = new RectangleF(10, 10, 200, 30);
//Set text
textField.Text = "สวัสดิ์ชาวโลก";
//Load the font as stream
Stream fontStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
tahoma.ttf");
//Create a new PDF font instance
PdfFont font = new PdfTrueTypeFont(fontStream, 10);
//Set font
textField.Font = font;
//Add the text box field to the form collection
document.Form.Fields.Add(textField);
//Set default appearance as false
document.Form.SetDefaultAppearance(false);
//Enable complex script layout for form
document.Form.ComplexScript = true;
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the PDF document
document.Close(true);
//Save the stream into PDF file
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

You can also flatten the existing form fields with complex script layout by using the following code snippet.

C#

```
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Form.pdf");
//Get the existing PDF form
PdfLoadedForm lForm = loadedDocument.Form as PdfLoadedForm;
//Set the complex script layout
lForm.ComplexScript = true;
//Set flatten
lForm.Flatten = true;
//Save the document
loadedDocument.Save("flatten.pdf");
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document
Dim loadedDocument As New PdfLoadedDocument("Form.pdf")
'Get the existing PDF form
Dim lForm As PdfLoadedForm = TryCast(loadedDocument.Form, PdfLoadedForm)
'Set the complex script layout
lForm.ComplexScript = True
'Set flatten
lForm.Flatten = True
'Save the document
loadedDocument.Save("flatten.pdf")
'Close the document
loadedDocument.Close(True)
```

UWP

```
Stream inputFileStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Form.pdf"
);
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputFileStream);
```

```
//Get the existing PDF form
PdfLoadedForm lForm = loadedDocument.Form as PdfLoadedForm;
//Set the complex script layout
lForm.ComplexScript = true;
//Set flatten
lForm.Flatten = true;
MemoryStream stream = new MemoryStream();
//Save the document
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
FileStream inputFileStream = new FileStream("Form.pdf", FileMode.Open,
FileAccess.Read);
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputFileStream);
//Get the existing PDF form
PdfLoadedForm lForm = loadedDocument.Form as PdfLoadedForm;
//Set the complex script layout
lForm.ComplexScript = true;
//Set flatten
lForm.Flatten = true;
MemoryStream stream = new MemoryStream();
//Save the document
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close(true);
//Defining the content type for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
Stream inputFileStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Form.pdf");
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputFileStream);
//Get the existing PDF form
PdfLoadedForm lForm = loadedDocument.Form as PdfLoadedForm;
//Set the complex script layout
lForm.ComplexScript = true;
//Set flatten
lForm.Flatten = true;
//Save the PDF document
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
```

```
//Close the PDF document
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Auto naming of form fields

The Essential PDF supports auto naming of form fields in a PDF document while creating form fields with same name. The [FieldAutoNaming](#) property of [PdfForm](#) is used to enable or disable auto naming of form field.

While enabling this property, the field names are auto naming. If the fields are created using same/common name, the created fields will act as individual.

While disabling this property, the field names are not auto naming and the created fields are saved in a single group. The same value will be referred in all the same name fields.

By default, the value is set to true. This is illustrated in the following code sample.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new page to the PDF document
PdfPage page = document.Pages.Add();
//Create the form
PdfForm form = document.Form;
//Enable the field auto naming
form.FieldAutoNaming = true;
//Create a textbox field and add the properties
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "Name");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "FirstName";
textBoxField.Text = "John";
//Add the form field to the document
document.Form.Fields.Add(textBoxField);
//Create a text box field with the same name and add the properties
PdfTextBoxField textBoxField1 = new PdfTextBoxField(page, "Name");
textBoxField1.Bounds = new RectangleF(0, 50, 100, 20);
textBoxField1.ToolTip = "LastName";
textBoxField1.Text = "Doe";
//Add form field to the document
document.Form.Fields.Add(textBoxField1);
//Save the document
```



```
document.Save("Output.pdf");
//Close the document
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As New PdfDocument()
'Add a new page to the PDF document
Dim page As PdfPage = document.Pages.Add()
'Create the form
Dim form As PdfForm = document.Form
'Enable the field auto naming
form.FieldAutoNaming = True
'Create a text box field and add the properties
Dim textBoxField As New PdfTextBoxField(page, "Name")
textBoxField.Bounds = New RectangleF(0, 0, 100, 20)
textBoxField.ToolTip = "FirstName"
textBoxField.Text = "John"
'Add the form field to the document
document.Form.Fields.Add(textBoxField)
'Create a text box field with the same name and add the properties
Dim textBoxField1 As New PdfTextBoxField(page, "Name")
textBoxField1.Bounds = New RectangleF(0, 50, 100, 20)
textBoxField1.ToolTip = "LastName"
textBoxField1.Text = "Doe"
'Add form field to the document
document.Form.Fields.Add(textBoxField1)
'Save the document
document.Save("Output.pdf")
'Close the document
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new page to the PDF document
PdfPage page = document.Pages.Add();
//Create the form
PdfForm form = document.Form;
//Enable the field auto naming
form.FieldAutoNaming = true;
//Create a text box field and add the properties
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "Name");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "FirstName";
textBoxField.Text = "John";
//Add the form field to the document
document.Form.Fields.Add(textBoxField);
//Create a text box field with the same name and add the properties
PdfTextBoxField textBoxField1 = new PdfTextBoxField(page, "Name");
textBoxField1.Bounds = new RectangleF(0, 50, 100, 20);
textBoxField1.ToolTip = "LastName";
textBoxField1.Text = "Doe";
//Add form field to the document
```

```
document.Form.Fields.Add(textBoxField1);
//Save the document as stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new page to the PDF document
PdfPage page = document.Pages.Add();
//Create the form
PdfForm form = document.Form;
//Enable the field auto naming
form.FieldAutoNaming = true;
//Create a text box field and add the properties
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "Name");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "FirstName";
textBoxField.Text = "John";
//Add the form field to the document
document.Form.Fields.Add(textBoxField);
//Create a text box field with the same name and add the properties
PdfTextBoxField textBoxField1 = new PdfTextBoxField(page, "Name");
textBoxField1.Bounds = new RectangleF(0, 50, 100, 20);
textBoxField1.ToolTip = "LastName";
textBoxField1.Text = "Doe";
//Add form field to the document
document.Form.Fields.Add(textBoxField1);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file.
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

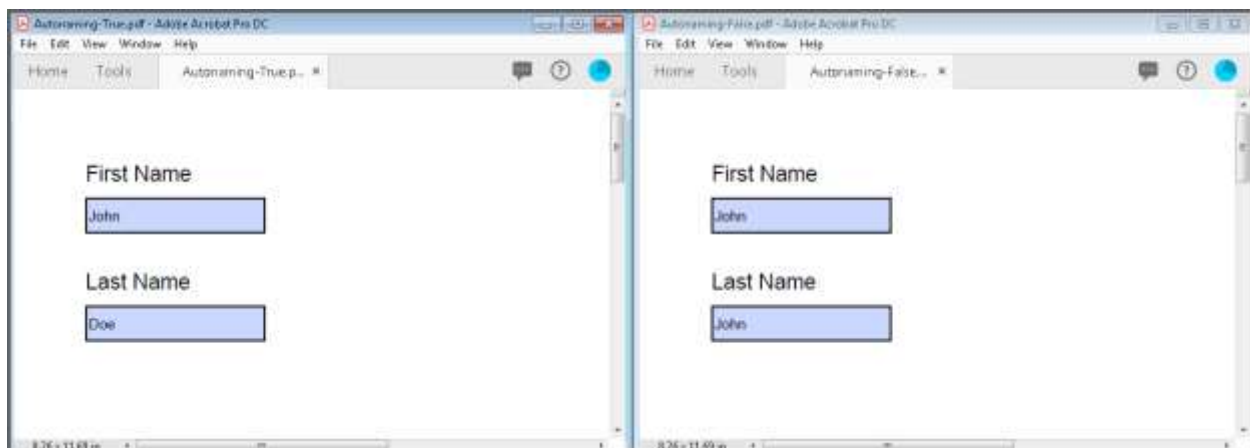
```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new page to the PDF document
PdfPage page = document.Pages.Add();
```

```

//Create the form
PdfForm form = document.Form;
//Enable the field auto naming
form.FieldAutoNaming = true;
//Create a textbox field and add the properties
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "Name");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "FirstName";
textBoxField.Text = "John";
//Add the form field to the document
document.Form.Fields.Add(textBoxField);
//Create a text box field with the same name and add the properties
PdfTextBoxField textBoxField1 = new PdfTextBoxField(page, "Name");
textBoxField1.Bounds = new RectangleF(0, 50, 100, 20);
textBoxField1.ToolTip = "LastName";
textBoxField1.Text = "Doe";
//Add form field to the document
document.Form.Fields.Add(textBoxField1);
//Save the document as stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following screenshot shows the enabling and disabling of form fields auto naming in PDF documents.



Adding actions to form fields

Please refer to the [actions](#) section for more details

Note: Essential PDF allows users to preserve the extended rights for form filling alone.

Auto resizing text box field text

The Essential PDF provides support to automatically resize the text of text box field based on the field width and height. You can auto resize the text box text by using the [AutoResizeText](#) property available in [PdfLoadedTextBox](#) instance.

The following code illustrates how to set `AutoResizeText` in an existing PDF text box field.

C#

```
//Load an existing document
PdfLoadedDocument doc = new PdfLoadedDocument("SourceForm.pdf");
//Read the text box field
PdfLoadedForm form = doc.Form;
PdfLoadedTextBoxField loadedField = doc.Form.Fields[0] as
PdfLoadedTextBoxField;
//Enable auto resize
loadedField.AutoResizeText = true;
//Flatten the form
form.Flatten = true;
//Save the document
doc.Save("Form.pdf");
//Close the document
doc.Close(true);
```

VB.NET

```
'Load an existing document
Dim doc As PdfLoadedDocument = New PdfLoadedDocument("SourceForm.pdf")
Dim form As PdfLoadedForm = doc.Form
'Read the text box field
Dim loadedField As PdfLoadedTextBoxField = TryCast(doc.Form.Fields(0),
PdfLoadedTextBoxField)
'Enable auto resize.
loadedField.AutoResizeText = True
'Flatten the form
form.Flatten = True
'Save the document
doc.Save("Form.pdf")
'Close the document
doc.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
```

```

//Loads or opens an existing PDF document through Open method of the PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Read the text box field
PdfLoadedTextBoxField loadedField = loadedDocument.Form.Fields[0] as PdfLoadedTextBoxField;
//Enable auto resize
loadedField.AutoResizeText = true;
//Flatten the form
loadedForm.Flatten = true;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the PDF/UWP section for respective code samples
Save(stream, "sample.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open, FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Read the text box field
PdfLoadedTextBoxField loadedField = loadedDocument.Form.Fields[0] as PdfLoadedTextBoxField;
//Enable auto resize
loadedField.AutoResizeText = true;
//Flatten the form
loadedForm.Flatten = true;
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.Sample.pdf");

```

```

PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form
PdfLoadedForm loadedForm = loadedDocument.Form;
//Read the text box field
PdfLoadedTextBoxField loadedField = loadedDocument.Form.Fields[0] as
PdfLoadedTextBoxField;
//Enable auto resize
loadedField.AutoResizeText = true;
//Flatten the form
form.Flatten = true;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
"application/pdf", stream);
}

```

Troubleshooting

Form fields may appear empty in adobe reader some time due to the absence of the appearance dictionary. To resolve this, you have to enable the Adobe Reader default appearance by using the [SetDefaultAppearance](#) method in [PdfForm](#) class.

The below code illustrates how to enable the default appearance in new PDF document.

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a Text box field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Add the form field to the document.
document.Form.Fields.Add(textBoxField);
//Enable the default Appearance
document.Form.SetDefaultAppearance(true);
//Save the document.
document.Save("Form.pdf");
//close the document
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
'Add a new page to PDF document.
Dim page As PdfPage = document.Pages.Add()
'Create a Text box field and add the properties.
Dim textBoxField As New PdfTextBoxField(page, "FirstName")
textBoxField.Bounds = New RectangleF(0, 0, 100, 20)
textBoxField.ToolTip = "First Name"
'Add the form field to the document.
document.Form.Fields.Add(textBoxField)
'Enable the default Appearance
document.Form.SetDefaultAppearance(True)
'Save the document.
document.Save("Form.pdf")
'close the document
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a new page to PDF document
PdfPage page = document.Pages.Add();
//Create a Text box field and add the properties
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Add the form field to the document
document.Form.Fields.Add(textBoxField);
//Enable the default Appearance
document.Form.SetDefaultAppearance(true);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Form.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a Text box field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Add the form field to the document.
document.Form.Fields.Add(textBoxField);

```

```
//Enable the default Appearance
document.Form.SetDefaultAppearance(true);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Form.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a new page to PDF document.
PdfPage page = document.Pages.Add();
//Create a Text box field and add the properties.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
textBoxField.Bounds = new Syncfusion.Drawing.RectangleF(0, 0, 100, 20);
textBoxField.ToolTip = "First Name";
//Add the form field to the document.
document.Form.Fields.Add(textBoxField);
//Enable the default Appearance
document.Form.SetDefaultAppearance(true);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Form.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Form.pdf",
"application/pdf", stream);
}
```

The below code illustrates how to enable the default appearance in existing PDF document.

C#

```
//Load the PDF document.
```



```

PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded text box field and fill it.
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "First Name";
//Enable the default Appearance
loadedDocument.Form.SetDefaultAppearance(true);
//Save the modified document.
loadedDocument.Save("sample.pdf");
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Get the loaded form.
Dim loadedForm As PdfLoadedForm = loadedDocument.Form
'Get the loaded text box field and fill it.
Dim loadedTextBoxField As PdfLoadedTextBoxField =
TryCast(loadedForm.Fields(0), PdfLoadedTextBoxField)
loadedTextBoxField.Text = "First Name"
'Enable the default Appearance
loadedDocument.Form.SetDefaultAppearance(True)
'Save the modified document.
loadedDocument.Save("sample.pdf")
'Close the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded text box field and fill it.
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "First Name";
//Enable the default Appearance
loadedDocument.Form.SetDefaultAppearance(true);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);

```

```
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples
Save(stream, "sample.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded text box field and fill it.
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "First Name";
//Enable the default Appearance
loadedDocument.Form.SetDefaultAppearance(true);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the loaded form.
PdfLoadedForm loadedForm = loadedDocument.Form;
//Get the loaded text box field and fill it.
PdfLoadedTextBoxField loadedTextBoxField = loadedForm.Fields[0] as
PdfLoadedTextBoxField;
loadedTextBoxField.Text = "First Name";
//Enable the default Appearance
loadedDocument.Form.SetDefaultAppearance(true);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

Working with XFA

XFA stands for XML Forms Architecture, dynamic forms are based on a XML specification. Essential PDF supports both the dynamic and static XFA forms.

- In a static form the form's appearance and layout is fixed, regardless of the field content.
- Dynamic forms can change in appearance in several ways in response to changes in the data.

Creating a new XFA form

Essential PDF allows you to create and manipulate the XFA form in PDF document by using [PdfXfaDocument](#) and [PdfXfaForm](#) classes. The [PdfXfaFieldCollection](#) class represents the entire field collection of the XFA form.

Adding a new page to the XFA document

The following code sample explains you on how to add a new page in a PDF XFA document.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!");
//Add the field to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//Close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form

```

```

Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a text element and add the properties
Dim textElement As New PdfXfaTextElement("Hello World!")
'Add the field to the XFA form
mainForm.Fields.Add(textElement)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'Close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!");
//Add the field to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!");
//Add the field to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty

```

```

stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!");
//Add the field to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Note: XFA documents are created based on the flow layout, so the pages are sequentially added if the doesn't have space to fit the field it will automatically move on the next page, so we can't specify the page and page numbers.

PDF supports XFA forms only in Windows Forms, WPF, ASP.NET, ASP.NET MVC, UWP platforms, Xamarin (.NETStandard 2.0 and above), and ASP.NET Core (.NETStandard 2.0 and above).

Adding the XFA document settings

Essential PDF supports various XFA page setting options through [PdfXfaPageSettings](#) class, to control the page display.

The below sample illustrates how to create a new PDF document with XFA page size.

C#

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = PdfPageSize.A4;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();
```

VB.NET

```
'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Set the page size
document.PageSettings.PageSize = PdfPageSize.A4
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a text element and add the properties
Dim textElement As New PdfXfaTextElement("Hello World!!!")
'Set font
textElement.Font = New PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold)
'Add the text element to the XFA form
mainForm.Fields.Add(textElement)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()
```

UWP

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = PdfPageSize.A4;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = PdfPageSize.A4;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document.
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
```

```
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = PdfPageSize.A4;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document.
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

You can create a custom page size to the PDF document by using following code snippet.

C#

```
//Create a new PDF XFA document
```



```

PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = new SizeF(500,700);
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1",xfaPage,xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf",PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Set the page size
document.PageSettings.PageSize = New SizeF(500,700)
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New
PdfXfaForm("subform1",xfaPage,xfaPage.GetClientSize().Width)
'Create a text element and add the properties
Dim textElement As New PdfXfaTextElement("Hello World!!!")
'Set font
textElement.Font = New PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold)
'Add the text element to the XFA form
mainForm.Fields.Add(textElement)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf",PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = new SizeF(500, 700);
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form

```

```

PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = new SizeF(500, 700);
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = new SizeF(500, 700);
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

You can change page orientation from portrait to landscape by using the following code snippet.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = new SizeF(500, 700);
//Change the page orientation to landscape
document.PageSettings.PageOrientation = PdfXfaPageOrientation.Landscape;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form

```

```

PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize ().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Set the page size
document.PageSettings.PageSize = New SizeF(500,700)
'Change the page orientation to landscape
document.PageSettings.PageOrientation = PdfXfaPageOrientation.Landscape
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a text element and add the properties
Dim textElement As New PdfXfaTextElement("Hello World!!!")
'Set font
textElement.Font = New PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold)
'Add the text element to the XFA form
mainForm.Fields.Add(textElement)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = new SizeF(500, 700);
//Change the page orientation to landscape
document.PageSettings.PageOrientation = PdfXfaPageOrientation.Landscape;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);

```

```

//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = new SizeF(500, 700);
//Change the page orientation to landscape
document.PageSettings.PageOrientation = PdfXfaPageOrientation.Landscape;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = new SizeF(500, 700);
//Change the page orientation to landscape
document.PageSettings.PageOrientation = PdfXfaPageOrientation.Landscape;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Creating dynamic XFA forms

The below sample illustrates how to create a dynamic XFA forms.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = PdfPageSize.A4;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();

```

```

//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize ().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document dynamic form
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Set the page size
document.PageSettings.PageSize = PdfPageSize.A4
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New PdfXfaForm("subform1",
xfaPage, xfaPage.GetClientSize().Width)
'Create a text element and add the properties
Dim textElement As New PdfXfaTextElement("Hello World!!!")
'Set font
textElement.Font = New PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold)
'Add the text element to the XFA form
mainForm.Fields.Add(textElement)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document dynamic form
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = PdfPageSize.A4;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font

```

```

textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = PdfPageSize.A4;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = PdfPageSize.A4;

```



```

//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Creating static XFA forms

The below sample illustrates how to create a static XFA forms.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = PdfPageSize.A4;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);

```

```
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document with static form
document.Save("XfaForm.pdf", PdfXfaType.Static);
//close the document
document.Close();
```

VB.NET

```
'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Set the page size
document.PageSettings.PageSize = PdfPageSize.A4
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a text element and add the properties
Dim textElement As New PdfXfaTextElement("Hello World!!!")
'Set font
textElement.Font = New PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold)
'Add the text element to the XFA form
mainForm.Fields.Add(textElement)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document with static form
document.Save("XfaForm.pdf", PdfXfaType.Static)
'close the document
document.Close()
```

UWP

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = PdfPageSize.A4;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
```

```
document.Save(stream, PdfXfaType.Static);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = PdfPageSize.A4;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Static);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Set the page size
document.PageSettings.PageSize = PdfPageSize.A4;
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!!!");
//Set font
```

```

textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Static);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Creating XFA form fields

[Adding the XFA text box field.](#)

The below code snippet illustrates how to add a textbox field to a new PDF document using [PdfXfaTextBoxField](#) class.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);

```

```
//close the document
document.Close();
```

VB.NET

```
'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a textbox field and add the properties
Dim textBoxField As New PdfXfaTextBoxField("FirstName", New SizeF(200, 20))
'Set the caption text
textBoxField.Caption.Text = "First Name"
'Set the tool tip
textBoxField.ToolTip = "First Name"
'Add the field to the XFA form
mainForm.Fields.Add(textBoxField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()
```

UWP

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
```

```

//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

The below code snippet illustrates how to add a textbox field to an existing PDF document

C#

```

//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new
PdfLoadedXfaDocument("XfaForm.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the existing XFA form
loadedForm.Fields.Add(textBoxField);
//Save the document
loadedDocument.Save("XfaForm.pdf");
//Close the document
loadedDocument.Close();

```

VB.NET

```

'Load the existing XFA document
Dim loadedDocument As New PdfLoadedXfaDocument("XfaForm.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Create a textbox field and add the properties
Dim textBoxField As New PdfXfaTextBoxField("FirstName", New SizeF(200, 20))
'Set the caption text
textBoxField.Caption.Text = "First Name"

```

```

'Set the tool tip
textBoxField.ToolTip = "First Name"
'Add the field to the existing XFA form
loadedForm.Fields.Add(textBoxField)
'Save the document
loadedDocument.Save("XfaForm.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.XfaForm.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the existing XFA form
loadedForm.Fields.Add(textBoxField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the existing XFA form
loadedForm.Fields.Add(textBoxField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);

```



```
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the existing XFA form
loadedForm.Fields.Add(textBoxField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Adding the XFA numeric field

The below code snippet illustrates how to add a numeric field to a new PDF document using [PdfXfaNumericField](#) class.

C#

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a numeric field and add the properties
PdfXfaNumericField numericField = new PdfXfaNumericField("numericField", new
SizeF(200, 20));
//Set the caption text
numericField.Caption.Text = "Numeric Field";
//Add the field to the XFA form
mainForm.Fields.Add(numericField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();
```

VB.NET

```
'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a numeric field and add the properties
Dim numericField As New PdfXfaNumericField("numericField", New SizeF(200, 20))
'Set the caption text
numericField.Caption.Text = "Numeric Field"
'Add the field to the XFA form
mainForm.Fields.Add(numericField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()
```

UWP

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
```

```

PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a numeric field and add the properties
PdfXfaNumericField numericField = new PdfXfaNumericField("numericField", new
SizeF(200, 20));
//Set the caption text
numericField.Caption.Text = "Numeric Field";
//Add the field to the XFA form
mainForm.Fields.Add(numericField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a numeric field and add the properties
PdfXfaNumericField numericField = new PdfXfaNumericField("numericField", new
SizeF(200, 20));
//Set the caption text
numericField.Caption.Text = "Numeric Field";
//Add the field to the XFA form
mainForm.Fields.Add(numericField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document

```

```

PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a numeric field and add the properties
PdfXfaNumericField numericField = new PdfXfaNumericField("numericField", new
SizeF(200, 20));
//Set the caption text
numericField.Caption.Text = "Numeric Field";
//Add the field to the XFA form
mainForm.Fields.Add(numericField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

The below code snippet illustrates how to add the numeric field to an existing PDF document.

C#

```

//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new
PdfLoadedXfaDocument("XfaForm.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a numeric field and add the properties
PdfXfaNumericField numericField = new PdfXfaNumericField("numericField", new
SizeF(200, 20));
//Set the caption text
numericField.Caption.Text = "Numeric Field";
//Add the field to the existing XFA form
loadedForm.Fields.Add(numericField);
//Save the document
loadedDocument.Save("XfaForm.pdf");

```

```
//Close the document
loadedDocument.Close();
```

VB.NET

```
'Load the existing XFA document
Dim loadedDocument As New PdfLoadedXfaDocument("XfaForm.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Create a numeric field and add the properties
Dim numericField As New PdfXfaNumericField("numericField", New SizeF(200, 20))
'Set the caption text
numericField.Caption.Text = "Numeric Field"
'Add the field to the existing XFA form
loadedForm.Fields.Add(numericField)
'Save the document
loadedDocument.Save("XfaForm.pdf")
'Close the document
loadedDocument.Close()
```

UWP

```
//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.XfaForm.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a numeric field and add the properties
PdfXfaNumericField numericField = new PdfXfaNumericField("numericField", new
SizeF(200, 20));
//Set the caption text
numericField.Caption.Text = "Numeric Field";
//Add the field to the existing XFA form
loadedForm.Fields.Add(numericField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a numeric field and add the properties
```

```

PdfXfaNumericField numericField = new PdfXfaNumericField("numericField", new
SizeF(200, 20));
//Set the caption text
numericField.Caption.Text = "Numeric Field";
//Add the field to the existing XFA form
loadedForm.Fields.Add(numericField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a numeric field and add the properties
PdfXfaNumericField numericField = new PdfXfaNumericField("numericField", new
SizeF(200, 20));
//Set the caption text
numericField.Caption.Text = "Numeric Field";
//Add the field to the existing XFA form
loadedForm.Fields.Add(numericField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{

```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Adding the XFA combo box field

The below code snippet illustrates how to add a combo box field to a new PDF document using [PdfXfaComboBoxField](#) class.

C#

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a combo box field and add the properties
PdfXfaComboBoxField comboBoxField = new PdfXfaComboBoxField("comboBoxField",
new SizeF(200, 20));
//Set the caption text
comboBoxField.Caption.Text = "Job Title";
//Add the combo box items
comboBoxField.Items.Add("Development.");
comboBoxField.Items.Add("Support.");
comboBoxField.Items.Add("Documentation.");
//Add the field to the XFA form
mainForm.Fields.Add(comboBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();
```

VB.NET

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a combo box field and add the properties
PdfXfaComboBoxField comboBoxField = new PdfXfaComboBoxField("comboBoxField",
new SizeF(200, 20));
//Set the caption text
comboBoxField.Caption.Text = "Job Title";
//Add the combo box items
comboBoxField.Items.Add("Development.");
comboBoxField.Items.Add("Support.");
comboBoxField.Items.Add("Documentation.");
//Add the field to the XFA form
mainForm.Fields.Add(comboBoxField);
```

```
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();
```

UWP

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a combo box field and add the properties
PdfXfaComboBoxField comboBoxField = new PdfXfaComboBoxField("comboBoxField",
new SizeF(200, 20));
//Set the caption text
comboBoxField.Caption.Text = "Job Title";
//Add the combo box items
comboBoxField.Items.Add("Development.");
comboBoxField.Items.Add("Support.");
comboBoxField.Items.Add("Documentation.");
//Add the field to the XFA form
mainForm.Fields.Add(comboBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a combo box field and add the properties
PdfXfaComboBoxField comboBoxField = new PdfXfaComboBoxField("comboBoxField",
new SizeF(200, 20));
//Set the caption text
comboBoxField.Caption.Text = "Job Title";
//Add the combo box items
comboBoxField.Items.Add("Development.");
comboBoxField.Items.Add("Support.");
comboBoxField.Items.Add("Documentation.");
//Add the field to the XFA form
```



```

mainForm.Fields.Add(comboBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a combo box field and add the properties
PdfXfaComboBoxField comboBoxField = new PdfXfaComboBoxField("comboBoxField",
new SizeF(200, 20));
//Set the caption text
comboBoxField.Caption.Text = "Job Title";
//Add the combo box items
comboBoxField.Items.Add("Development.");
comboBoxField.Items.Add("Support.");
comboBoxField.Items.Add("Documentation.");
//Add the field to the XFA form
mainForm.Fields.Add(comboBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else

```

```
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
    "application/pdf", fdfStream);
}
```

The below code snippet illustrates how to add the combo box field to an existing PDF document.

C#

```
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new
PdfLoadedXfaDocument("XfaForm.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a combo box field and add the properties
PdfXfaComboBoxField comboBoxField = new PdfXfaComboBoxField("comboBoxField",
new SizeF(200, 20));
//Set the caption text
comboBoxField.Caption.Text = "Job Title";
//Add the combo box items
comboBoxField.Items.Add("Development.");
comboBoxField.Items.Add("Support.");
comboBoxField.Items.Add("Documentation.");
//Add the field to the existing XFA form
loadedForm.Fields.Add(comboBoxField);
//Save the document
loadedDocument.Save("XfaForm.pdf");
//Close the document
loadedDocument.Close();
```

VB.NET

```
'Load the existing XFA document
Dim loadedDocument As New PdfLoadedXfaDocument("XfaForm.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Create a combo box field and add the properties
Dim comboBoxField As New PdfXfaComboBoxField("comboBoxField", New SizeF(200,
20))
'Set the caption text
comboBoxField.Caption.Text = "Job Title"
'Add the combo box items
comboBoxField.Items.Add("Development.")
comboBoxField.Items.Add("Support.")
comboBoxField.Items.Add("Documentation.")
'Add the field to the existing XFA form
loadedForm.Fields.Add(comboBoxField)
'Save the document
loadedDocument.Save("XfaForm.pdf")
'Close the document
loadedDocument.Close()
```

UWP

```
//Load the PDF document as stream
```

```

Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.XfaForm.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a combo box field and add the properties
PdfXfaComboBoxField comboBoxField = new PdfXfaComboBoxField("comboBoxField",
new SizeF(200, 20));
//Set the caption text
comboBoxField.Caption.Text = "Job Title";
//Add the combo box items
comboBoxField.Items.Add("Development.");
comboBoxField.Items.Add("Support.");
comboBoxField.Items.Add("Documentation.");
//Add the field to the existing XFA form
loadedForm.Fields.Add(comboBoxField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a combo box field and add the properties
PdfXfaComboBoxField comboBoxField = new PdfXfaComboBoxField("comboBoxField",
new SizeF(200, 20));
//Set the caption text
comboBoxField.Caption.Text = "Job Title";
//Add the combo box items
comboBoxField.Items.Add("Development.");
comboBoxField.Items.Add("Support.");
comboBoxField.Items.Add("Documentation.");
//Add the field to the existing XFA form
loadedForm.Fields.Add(comboBoxField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";

```

```
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a combo box field and add the properties
PdfXfaComboBoxField comboBoxField = new PdfXfaComboBoxField("comboBoxField",
new SizeF(200, 20));
//Set the caption text
comboBoxField.Caption.Text = "Job Title";
//Add the combo box items
comboBoxField.Items.Add("Development.");
comboBoxField.Items.Add("Support.");
comboBoxField.Items.Add("Documentation.");
//Add the field to the existing XFA form
loadedForm.Fields.Add(comboBoxField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Adding the XFA list box field

The below code snippet illustrates how to add a list box field to a new PDF document using [PdfXfaListBoxField](#) class.

C#

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
```

```

//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a list box field and add the properties
PdfXfaListBoxField listBoxField = new PdfXfaListBoxField("listBoxField", new
SizeF(150, 50));
//Set the caption text
listBoxField.Caption.Text = "Known Languages";
listBoxField.Caption.Position = PdfXfaPosition.Top;
listBoxField.Caption.HorizontalAlignment = PdfXfaHorizontalAlignment.Center;
//Add the list box items
listBoxField.Items.Add("English");
listBoxField.Items.Add("French");
listBoxField.Items.Add("German");
//Add the field to the XFA form
mainForm.Fields.Add(listBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a list box field and add the properties
Dim listBoxField As New PdfXfaListBoxField("listBoxField", New SizeF(150,
50))
'Set the caption text
listBoxField.Caption.Text = "Known Languages"
listBoxField.Caption.Position = PdfXfaPosition.Top
listBoxField.Caption.HorizontalAlignment = PdfXfaHorizontalAlignment.Center
'Add the list box items
listBoxField.Items.Add("English")
listBoxField.Items.Add("French")
listBoxField.Items.Add("German")
'Add the field to the XFA form
mainForm.Fields.Add(listBoxField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a list box field and add the properties
PdfXfaListBoxField listBoxField = new PdfXfaListBoxField("listBoxField", new
SizeF(150, 50));
//Set the caption text
listBoxField.Caption.Text = "Known Languages";
listBoxField.Caption.Position = PdfXfaPosition.Top;
listBoxField.Caption.HorizontalAlignment = PdfXfaHorizontalAlignment.Center;
//Add the list box items
listBoxField.Items.Add("English");
listBoxField.Items.Add("French");
listBoxField.Items.Add("German");
//Add the field to the XFA form
mainForm.Fields.Add(listBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a list box field and add the properties
PdfXfaListBoxField listBoxField = new PdfXfaListBoxField("listBoxField", new
SizeF(150, 50));
//Set the caption text
listBoxField.Caption.Text = "Known Languages";
listBoxField.Caption.Position = PdfXfaPosition.Top;
listBoxField.Caption.HorizontalAlignment = PdfXfaHorizontalAlignment.Center;
//Add the list box items
listBoxField.Items.Add("English");
listBoxField.Items.Add("French");
listBoxField.Items.Add("German");
//Add the field to the XFA form
mainForm.Fields.Add(listBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);

```

```
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.ClientSize().Width);
//Create a list box field and add the properties
PdfXfaListBoxField listBoxField = new PdfXfaListBoxField("listBoxField", new
SizeF(150, 50));
//Set the caption text
listBoxField.Caption.Text = "Known Languages";
listBoxField.Caption.Position = PdfXfaPosition.Top;
listBoxField.Caption.HorizontalAlignment = PdfXfaHorizontalAlignment.Center;
//Add the list box items
listBoxField.Items.Add("English");
listBoxField.Items.Add("French");
listBoxField.Items.Add("German");
//Add the field to the XFA form
mainForm.Fields.Add(listBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

```
}

```

The below code snippet illustrates how to add the list box field to an existing PDF document.

C#

```
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new
PdfLoadedXfaDocument("XfaForm.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a list box field and add the properties
PdfXfaListBoxField listBoxField = new PdfXfaListBoxField("listBoxField", new
SizeF(150, 50));
//Set the caption text
listBoxField.Caption.Text = "Known Languages";
listBoxField.Caption.Position = PdfXfaPosition.Top;
listBoxField.Caption.HorizontalAlignment = PdfXfaHorizontalAlignment.Center;
//Add the list box items
listBoxField.Items.Add("English");
listBoxField.Items.Add("French");
listBoxField.Items.Add("German");
//Add the field to the existing XFA form
loadedForm.Fields.Add(listBoxField);
//Save the document
loadedDocument.Save("XfaForm.pdf");
//Close the document
loadedDocument.Close();
```

VB.NET

```
'Load the existing XFA document
Dim loadedDocument As New PdfLoadedXfaDocument("XfaForm.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Create a list box field and add the properties
Dim listBoxField As New PdfXfaListBoxField("listBoxField", New SizeF(150,
50))
'Set the caption text
listBoxField.Caption.Text = "Known Languages"
listBoxField.Caption.Position = PdfXfaPosition.Top
listBoxField.Caption.HorizontalAlignment = PdfXfaHorizontalAlignment.Center
'Add the list box items
listBoxField.Items.Add("English")
listBoxField.Items.Add("French")
listBoxField.Items.Add("German")
'Add the field to the existing XFA form
loadedForm.Fields.Add(listBoxField)
'Save the document
loadedDocument.Save("XfaForm.pdf")
'Close the document
loadedDocument.Close()
```

UWP


```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.XfaForm.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a list box field and add the properties
PdfXfaListBoxField listBoxField = new PdfXfaListBoxField("listBoxField", new
SizeF(150, 50));
//Set the caption text
listBoxField.Caption.Text = "Known Languages";
listBoxField.Caption.Position = PdfXfaPosition.Top;
listBoxField.Caption.HorizontalAlignment = PdfXfaHorizontalAlignment.Center;
//Add the list box items
listBoxField.Items.Add("English");
listBoxField.Items.Add("French");
listBoxField.Items.Add("German");
//Add the field to the existing XFA form
loadedForm.Fields.Add(listBoxField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a list box field and add the properties
PdfXfaListBoxField listBoxField = new PdfXfaListBoxField("listBoxField", new
SizeF(150, 50));
//Set the caption text
listBoxField.Caption.Text = "Known Languages";
listBoxField.Caption.Position = PdfXfaPosition.Top;
listBoxField.Caption.HorizontalAlignment = PdfXfaHorizontalAlignment.Center;
//Add the list box items
listBoxField.Items.Add("English");
listBoxField.Items.Add("French");
listBoxField.Items.Add("German");
//Add the field to the existing XFA form
loadedForm.Fields.Add(listBoxField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document

```

```
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a list box field and add the properties
PdfXfaListBoxField listBoxField = new PdfXfaListBoxField("listBoxField", new
SizeF(150, 50));
//Set the caption text
listBoxField.Caption.Text = "Known Languages";
listBoxField.Caption.Position = PdfXfaPosition.Top;
listBoxField.Caption.HorizontalAlignment = PdfXfaHorizontalAlignment.Center;
//Add the list box items
listBoxField.Items.Add("English");
listBoxField.Items.Add("French");
listBoxField.Items.Add("German");
//Add the field to the existing XFA form
loadedForm.Fields.Add(listBoxField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Adding XFA check box field

The below code snippet illustrates how to add a check box field to a new PDF document using [PdfXfaCheckBoxField](#) class.

C#

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a check box field and add the properties
PdfXfaCheckBoxField checkBoxField = new PdfXfaCheckBoxField("checkBoxField",
new SizeF(100, 20));
//Set the caption text
checkBoxField.Caption.Text = "Check box Field";
checkBoxField.CheckedStyle = PdfXfaCheckedStyle.Cross;
//Add the field to the XFA form
mainForm.Fields.Add(checkBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();
```

VB.NET

```
'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a check box field and add the properties
Dim checkBoxField As New PdfXfaCheckBoxField("checkBoxField", New SizeF(100,
20))
'Set the caption text
checkBoxField.Caption.Text = "Check box Field"
checkBoxField.CheckedStyle = PdfXfaCheckedStyle.Cross
'Add the field to the XFA form
mainForm.Fields.Add(checkBoxField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()
```

UWP

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
```

```

//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a check box field and add the properties
PdfXfaCheckBoxField checkBoxField = new PdfXfaCheckBoxField("checkBoxField",
new SizeF(100, 20));
//Set the caption text
checkBoxField.Caption.Text = "Check box Field";
checkBoxField.CheckedStyle = PdfXfaCheckedStyle.Cross;
//Add the field to the XFA form
mainForm.Fields.Add(checkBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a check box field and add the properties
PdfXfaCheckBoxField checkBoxField = new PdfXfaCheckBoxField("checkBoxField",
new SizeF(100, 20));
//Set the caption text
checkBoxField.Caption.Text = "Check box Field";
checkBoxField.CheckedStyle = PdfXfaCheckedStyle.Cross;
//Add the field to the XFA form
mainForm.Fields.Add(checkBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a check box field and add the properties
PdfXfaCheckBoxField checkBoxField = new PdfXfaCheckBoxField("checkBoxField",
new SizeF(100, 20));
//Set the caption text
checkBoxField.Caption.Text = "Check box Field";
checkBoxField.CheckedStyle = PdfXfaCheckedStyle.Cross;
//Add the field to the XFA form
mainForm.Fields.Add(checkBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

The below code snippet illustrates how to add the check box field to an existing PDF document.

C#

```

//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new
PdfLoadedXfaDocument("XfaForm.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a check box field and add the properties
PdfXfaCheckBoxField checkBoxField = new PdfXfaCheckBoxField("checkBoxField",
new SizeF(100, 20));
//Set the caption text
checkBoxField.Caption.Text = "Check box Field";
checkBoxField.CheckedStyle = PdfXfaCheckedStyle.Cross;
//Add the field to the existing XFA form

```

```
loadedForm.Fields.Add(checkBoxField);
//Save the document
loadedDocument.Save("XfaForm.pdf");
//Close the document
loadedDocument.Close();
```

VB.NET

```
'Load the existing XFA document
Dim loadedDocument As New PdfLoadedXfaDocument("XfaForm.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Create a check box field and add the properties
Dim checkBoxField As New PdfXfaCheckBoxField("checkBoxField", New SizeF(100,
20))
'Set the caption text
checkBoxField.Caption.Text = "Check box Field"
checkBoxField.CheckedStyle = PdfXfaCheckedStyle.Cross
'Add the field to the existing XFA form
loadedForm.Fields.Add(checkBoxField)
'Save the document
loadedDocument.Save("XfaForm.pdf")
'Close the document
loadedDocument.Close()
```

UWP

```
//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.XfaForm.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a check box field and add the properties
PdfXfaCheckBoxField checkBoxField = new PdfXfaCheckBoxField("checkBoxField",
new SizeF(100, 20));
//Set the caption text
checkBoxField.Caption.Text = "Check box Field";
checkBoxField.CheckedStyle = PdfXfaCheckedStyle.Cross;
//Add the field to the existing XFA form
loadedForm.Fields.Add(checkBoxField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Load the PDF document
```

```

FileStream docStream = new FileStream(fileName, FileMode.Open,
    FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a check box field and add the properties
PdfXfaCheckBoxField checkBoxField = new PdfXfaCheckBoxField("checkBoxField",
    new SizeF(100, 20));
//Set the caption text
checkBoxField.Caption.Text = "Check box Field";
checkBoxField.CheckedStyle = PdfXfaCheckedStyle.Cross;
//Add the field to the existing XFA form
loadedForm.Fields.Add(checkBoxField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the PDF document
Stream docStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
    input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a check box field and add the properties
PdfXfaCheckBoxField checkBoxField = new PdfXfaCheckBoxField("checkBoxField",
    new SizeF(100, 20));
//Set the caption text
checkBoxField.Caption.Text = "Check box Field";
checkBoxField.CheckedStyle = PdfXfaCheckedStyle.Cross;
//Add the field to the existing XFA form
loadedForm.Fields.Add(checkBoxField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Adding the XFA radio button field

The below code snippet illustrates how to add a radio button field to a new PDF document using [PdfXfaRadioButtonField](#) class and add it into [PdfXfaRadioButtonGroup](#).

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a radio button group
PdfXfaRadioButtonGroup group = new PdfXfaRadioButtonGroup("radioGroup");
group.FlowDirection = PdfXfaFlowDirection.Vertical;
//Create a radio button field and add the properties
PdfXfaRadioButtonField radioButtonField1 = new PdfXfaRadioButtonField("r1",
new.SizeF(80, 20));
//Set the caption text
radioButtonField1.Caption.Text = "Male";
PdfXfaRadioButtonField radioButtonField2 = new PdfXfaRadioButtonField("r2",
new.SizeF(80, 20));
radioButtonField2.Caption.Text = "Female";
//Add the radio button fields to the radio button group
group.Items.Add(radioButtonField1);
group.Items.Add(radioButtonField2);
//Add the field to the XFA form
mainForm.Fields.Add(group);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form

```



```

Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a radio button group
Dim group As New PdfXfaRadioButtonGroup("radioGroup")
group.FlowDirection = PdfXfaFlowDirection.Vertical
'Create a radio button field and add the properties
Dim radioButtonField1 As New PdfXfaRadioButtonField("r1", New SizeF(80, 20))
'Set the caption text
radioButtonField1.Caption.Text = "Male"
Dim radioButtonField2 As New PdfXfaRadioButtonField("r2", New SizeF(80, 20))
radioButtonField2.Caption.Text = "Female"
'Add the radio button fields to the radio button group
group.Items.Add(radioButtonField1)
group.Items.Add(radioButtonField2)
'Add the field to the XFA form
mainForm.Fields.Add(group)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a radio button group
PdfXfaRadioButtonGroup group = new PdfXfaRadioButtonGroup("radioGroup");
group.FlowDirection = PdfXfaFlowDirection.Vertical;
//Create a radio button field and add the properties
PdfXfaRadioButtonField radioButtonField1 = new PdfXfaRadioButtonField("r1",
new SizeF(80, 20));
//Set the caption text
radioButtonField1.Caption.Text = "Male";
PdfXfaRadioButtonField radioButtonField2 = new PdfXfaRadioButtonField("r2",
new SizeF(80, 20));
radioButtonField2.Caption.Text = "Female";
//Add the radio button fields to the radio button group
group.Items.Add(radioButtonField1);
group.Items.Add(radioButtonField2);
//Add the field to the XFA form
mainForm.Fields.Add(group);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples

```

```
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a radio button group
PdfXfaRadioButtonGroup group = new PdfXfaRadioButtonGroup("radioGroup");
group.FlowDirection = PdfXfaFlowDirection.Vertical;
//Create a radio button field and add the properties
PdfXfaRadioButtonField radioButtonField1 = new PdfXfaRadioButtonField("r1",
new SizeF(80, 20));
//Set the caption text
radioButtonField1.Caption.Text = "Male";
PdfXfaRadioButtonField radioButtonField2 = new PdfXfaRadioButtonField("r2",
new SizeF(80, 20));
radioButtonField2.Caption.Text = "Female";
//Add the radio button fields to the radio button group
group.Items.Add(radioButtonField1);
group.Items.Add(radioButtonField2);
//Add the field to the XFA form
mainForm.Fields.Add(group);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a radio button group
PdfXfaRadioButtonGroup group = new PdfXfaRadioButtonGroup("radioGroup");
group.FlowDirection = PdfXfaFlowDirection.Vertical;
//Create a radio button field and add the properties
```

```

PdfXfaRadioButtonField radioButtonField1 = new PdfXfaRadioButtonField("r1",
new SizeF(80, 20));
//Set the caption text
radioButtonField1.Caption.Text = "Male";
PdfXfaRadioButtonField radioButtonField2 = new PdfXfaRadioButtonField("r2",
new SizeF(80, 20));
radioButtonField2.Caption.Text = "Female";
//Add the radio button fields to the radio button group
group.Items.Add(radioButtonField1);
group.Items.Add(radioButtonField2);
//Add the field to the XFA form
mainForm.Fields.Add(group);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

The below code snippet illustrates how to add the radio button field to an existing PDF document.

C#

```

//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new
PdfLoadedXfaDocument("XfaForm.pdf");
//Load the existing XFA form.
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a radio button group
PdfXfaRadioButtonGroup group = new PdfXfaRadioButtonGroup("radioGroup");
group.FlowDirection = PdfXfaFlowDirection.Vertical;
//Create a radio button field and add the properties
PdfXfaRadioButtonField radioButtonField1 = new PdfXfaRadioButtonField("r1",
new SizeF(80, 20));
//Set the caption text
radioButtonField1.Caption.Text = "Male";
PdfXfaRadioButtonField radioButtonField2 = new PdfXfaRadioButtonField("r2",
new SizeF(80, 20));
radioButtonField2.Caption.Text = "Female";

```

```

//Add the radio button fields to the radio button group
group.Items.Add(radioButtonField1);
group.Items.Add(radioButtonField2);
//Add the field to the existing XFA form
loadedForm.Fields.Add(group);
//Save the document
loadedDocument.Save("XfaForm.pdf");
//Close the document
loadedDocument.Close();

```

VB.NET

```

'Load the existing XFA document
Dim loadedDocument As New PdfLoadedXfaDocument("XfaForm.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Create a radio button group
Dim group As New PdfXfaRadioButtonGroup("radioGroup")
group.FlowDirection = PdfXfaFlowDirection.Vertical
'Create a radio button field and add the properties
Dim radioButtonField1 As New PdfXfaRadioButtonField("r1", New SizeF(80, 20))
'Set the caption text
radioButtonField1.Caption.Text = "Male"
Dim radioButtonField2 As New PdfXfaRadioButtonField("r2", New SizeF(80, 20))
radioButtonField2.Caption.Text = "Female"
'Add the radio button fields to the radio button group
group.Items.Add(radioButtonField1)
group.Items.Add(radioButtonField2)
'Add the field to the existing XFA form
loadedForm.Fields.Add(group)
'Save the document
loadedDocument.Save("XfaForm.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.XfaForm.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a radio button group
PdfXfaRadioButtonGroup group = new PdfXfaRadioButtonGroup("radioGroup");
group.FlowDirection = PdfXfaFlowDirection.Vertical;
//Create a radio button field and add the properties
PdfXfaRadioButtonField radioButtonField1 = new PdfXfaRadioButtonField("r1",
new SizeF(80, 20));
//Set the caption text
radioButtonField1.Caption.Text = "Male";
PdfXfaRadioButtonField radioButtonField2 = new PdfXfaRadioButtonField("r2",
new SizeF(80, 20));
radioButtonField2.Caption.Text = "Female";

```

```

//Add the radio button fields to the radio button group
group.Items.Add(radioButtonField1);
group.Items.Add(radioButtonField2);
//Add the field to the existing XFA form
loadedForm.Fields.Add(group);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a radio button group
PdfXfaRadioButtonGroup group = new PdfXfaRadioButtonGroup("radioGroup");
group.FlowDirection = PdfXfaFlowDirection.Vertical;
//Create a radio button field and add the properties
PdfXfaRadioButtonField radioButtonField1 = new PdfXfaRadioButtonField("r1",
new SizeF(80, 20));
//Set the caption text
radioButtonField1.Caption.Text = "Male";
PdfXfaRadioButtonField radioButtonField2 = new PdfXfaRadioButtonField("r2",
new SizeF(80, 20));
radioButtonField2.Caption.Text = "Female";
//Add the radio button fields to the radio button group
group.Items.Add(radioButtonField1);
group.Items.Add(radioButtonField2);
//Add the field to the existing XFA form
loadedForm.Fields.Add(group);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the PDF document

```

```

Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a radio button group
PdfXfaRadioButtonGroup group = new PdfXfaRadioButtonGroup("radioGroup");
group.FlowDirection = PdfXfaFlowDirection.Vertical;
//Create a radio button field and add the properties
PdfXfaRadioButtonField radioButtonField1 = new PdfXfaRadioButtonField("r1",
new SizeF(80, 20));
//Set the caption text
radioButtonField1.Caption.Text = "Male";
PdfXfaRadioButtonField radioButtonField2 = new PdfXfaRadioButtonField("r2",
new SizeF(80, 20));
radioButtonField2.Caption.Text = "Female";
//Add the radio button fields to the radio button group
group.Items.Add(radioButtonField1);
group.Items.Add(radioButtonField2);
//Add the field to the existing XFA form
loadedForm.Fields.Add(group);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

[Adding the XFA date time field](#)

The below code snippet illustrates how to add a date time field to a new PDF document using [PdfXfaDateTimeField](#) class.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form

```

```

PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize ().Width);
//Create a date time field and add the properties
PdfXfaDateTimeField dateTimeField = new PdfXfaDateTimeField("date", new
SizeF(200, 20));
//Set the caption text
dateTimeField.Caption.Text = "Date Time Field";
//Set the tool tip
dateTimeField.ToolTip = "Date Time";
//Add the field to the XFA form
mainForm.Fields.Add(dateTimeField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a date time field and add the properties
Dim dateTimeField As New PdfXfaDateTimeField("date", New SizeF(200, 20))
'Set the caption text
dateTimeField.Caption.Text = "Date Time Field"
'Set the tool tip
dateTimeField.ToolTip = "Date Time"
'Add the field to the XFA form
mainForm.Fields.Add(dateTimeField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a date time field and add the properties
PdfXfaDateTimeField dateTimeField = new PdfXfaDateTimeField("date", new
SizeF(200, 20));
//Set the caption text
dateTimeField.Caption.Text = "Date Time Field";

```

```
//Set the tool tip
dateTimeField.ToolTip = "Date Time";
//Add the field to the XFA form
mainForm.Fields.Add(dateTimeField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a date time field and add the properties
PdfXfaDateTimeField dateTimeField = new PdfXfaDateTimeField("date", new
SizeF(200, 20));
//Set the caption text
dateTimeField.Caption.Text = "Date Time Field";
//Set the tool tip
dateTimeField.ToolTip = "Date Time";
//Add the field to the XFA form
mainForm.Fields.Add(dateTimeField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
```



```

PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a date time field and add the properties
PdfXfaDateTimeField dateTimeField = new PdfXfaDateTimeField("date", new
SizeF(200, 20));
//Set the caption text
dateTimeField.Caption.Text = "Date Time Field";
//Set the tool tip
dateTimeField.ToolTip = "Date Time";
//Add the field to the XFA form
mainForm.Fields.Add(dateTimeField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

The below code snippet illustrates how to add the date time field to an existing PDF document.

C#

```

//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new
PdfLoadedXfaDocument("XfaForm.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a date time field and add the properties
PdfXfaDateTimeField dateTimeField = new PdfXfaDateTimeField("date", new
SizeF(200, 20));
//Set the caption text
dateTimeField.Caption.Text = "Date Time Field";
//Set the tool tip
dateTimeField.ToolTip = "Date Time";
//Add the field to the existing XFA form
loadedForm.Fields.Add(dateTimeField);
//Save the document
loadedDocument.Save("XfaForm.pdf");
//Close the document

```

```
loadedDocument.Close();
```

VB.NET

```
'Load the existing XFA document
Dim loadedDocument As New PdfLoadedXfaDocument("XfaForm.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Create a date time field and add the properties
Dim dateTimeField As New PdfXfaDateTimeField("date", New SizeF(200, 20))
'Set the caption text
dateTimeField.Caption.Text = "Date Time Field"
'Set the tool tip
dateTimeField.ToolTip = "Date Time"
'Add the field to the existing XFA form
loadedForm.Fields.Add(dateTimeField)
'Save the document
loadedDocument.Save("XfaForm.pdf")
'Close the document
loadedDocument.Close()
```

UWP

```
//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.XfaForm.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a date time field and add the properties
PdfXfaDateTimeField dateTimeField = new PdfXfaDateTimeField("date", new
SizeF(200, 20));
//Set the caption text
dateTimeField.Caption.Text = "Date Time Field";
//Set the tool tip
dateTimeField.ToolTip = "Date Time";
//Add the field to the existing XFA form
loadedForm.Fields.Add(dateTimeField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
```

```
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a date time field and add the properties
PdfXfaDateTimeField dateTimeField = new PdfXfaDateTimeField("date", new
SizeF(200, 20));
//Set the caption text
dateTimeField.Caption.Text = "Date Time Field";
//Set the tool tip
dateTimeField.ToolTip = "Date Time";
//Add the field to the existing XFA form
loadedForm.Fields.Add(dateTimeField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a date time field and add the properties
PdfXfaDateTimeField dateTimeField = new PdfXfaDateTimeField("date", new
SizeF(200, 20));
//Set the caption text
dateTimeField.Caption.Text = "Date Time Field";
//Set the tool tip
dateTimeField.ToolTip = "Date Time";
//Add the field to the existing XFA form
loadedForm.Fields.Add(dateTimeField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
```

```
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Adding the XFA button field

The below code snippet illustrates how to add a button field to a new PDF document using [PdfXfaButtonField](#) class.

C#

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a button field and add the properties
PdfXfaButtonField buttonField = new PdfXfaButtonField("buttonField", new
SizeF(70, 20));
//Set the caption text
buttonField.Content = "Click";
//Add the field to the XFA form
mainForm.Fields.Add(buttonField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();
```

VB.NET

```
'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a button field and add the properties
Dim buttonField As New PdfXfaButtonField("buttonField", New SizeF(70, 20))
'Set the caption text
buttonField.Content = "Click"
'Add the field to the XFA form
mainForm.Fields.Add(buttonField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
```

```
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()
```

UWP

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a button field and add the properties
PdfXfaButtonField buttonField = new PdfXfaButtonField("buttonField", new
SizeF(70, 20));
//Set the caption text
buttonField.Content = "Click";
//Add the field to the XFA form
mainForm.Fields.Add(buttonField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a button field and add the properties
PdfXfaButtonField buttonField = new PdfXfaButtonField("buttonField", new
SizeF(70, 20));
//Set the caption text
buttonField.Content = "Click";
//Add the field to the XFA form
mainForm.Fields.Add(buttonField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
```

```
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a button field and add the properties
PdfXfaButtonField buttonField = new PdfXfaButtonField("buttonField", new
SizeF(70, 20));
//Set the caption text
buttonField.Content = "Click";
//Add the field to the XFA form
mainForm.Fields.Add(buttonField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

The below code snippet illustrates how to add the button field to an existing PDF document.

C#

```
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new
PdfLoadedXfaDocument("XfaForm.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a button field and add the properties
```

```

PdfXfaButtonField buttonField = new PdfXfaButtonField("buttonField", new
SizeF(70, 20));
//Set the caption text
buttonField.Content = "Click";
//Add the field to the existing XFA form
loadedForm.Fields.Add(buttonField);
//Save the document
loadedDocument.Save("XfaForm.pdf");
//Close the document
loadedDocument.Close();

```

VB.NET

```

'Load the existing XFA document
Dim loadedDocument As New PdfLoadedXfaDocument("XfaForm.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Create a button field and add the properties
Dim buttonField As New PdfXfaButtonField("buttonField", New SizeF(70, 20))
'Set the caption text
buttonField.Content = "Click"
'Add the field to the existing XFA form
loadedForm.Fields.Add(buttonField)
'Save the document
loadedDocument.Save("XfaForm.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.XfaForm.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a button field and add the properties
PdfXfaButtonField buttonField = new PdfXfaButtonField("buttonField", new
SizeF(70, 20));
//Set the caption text
buttonField.Content = "Click";
//Add the field to the existing XFA form
loadedForm.Fields.Add(buttonField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
    FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a button field and add the properties
PdfXfaButtonField buttonField = new PdfXfaButtonField("buttonField", new
    SizeF(70, 20));
//Set the caption text
buttonField.Content = "Click";
//Add the field to the existing XFA form
loadedForm.Fields.Add(buttonField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
    input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a button field and add the properties
PdfXfaButtonField buttonField = new PdfXfaButtonField("buttonField", new
    SizeF(70, 20));
//Set the caption text
buttonField.Content = "Click";
//Add the field to the existing XFA form
loadedForm.Fields.Add(buttonField);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
```



```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Adding the XFA text element

The below code snippet illustrates how to add a text element to a new PDF document using [PdfXfaTextElement](#) class.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a text element and add the properties
Dim textElement As New PdfXfaTextElement("Hello World!")
'Set font
textElement.Font = New PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold)
'Add the text element to the XFA form
mainForm.Fields.Add(textElement)

```

```

'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document

```

```
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the XFA form
mainForm.Fields.Add(textElement);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

The below code snippet illustrates how to add a text element to an existing PDF document.

C#

```
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new
PdfLoadedXfaDocument("XfaForm.pdf");
```

```

//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the existing XFA form
loadedForm.Fields.Add(textElement);
//Save the document
loadedDocument.Save("XfaForm.pdf");
//Close the document
loadedDocument.Close();

```

VB.NET

```

'Load the existing XFA document
Dim loadedDocument As New PdfLoadedXfaDocument("XfaForm.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Create a text element and add the properties
Dim textElement As New PdfXfaTextElement("Hello World!")
'Set font
textElement.Font = New PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold)
'Add the text element to the existing XFA form
loadedForm.Fields.Add(textElement)
'Save the document
loadedDocument.Save("XfaForm.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.XfaForm.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the existing XFA form
loadedForm.Fields.Add(textElement);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the existing XFA form
loadedForm.Fields.Add(textElement);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a text element and add the properties
PdfXfaTextElement textElement = new PdfXfaTextElement("Hello World!");
//Set font
textElement.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 14,
PdfFontStyle.Bold);
//Add the text element to the existing XFA form
loadedForm.Fields.Add(textElement);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Adding the XFA rectangle field

The below code snippet illustrates how to add the rectangle field to a new PDF document using [PdfXfaRectangleField](#) class.

C#

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a rectangle field and add the properties
PdfXfaRectangleField rectangle = new PdfXfaRectangleField("rect1", new SizeF
(100, 50));
//Set the fill color
rectangle.Border.FillColor = new PdfXfaSolidBrush(Color.Red);
//Add the rectangle field to the XFA form
mainForm.Fields.Add(rectangle);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();
```

VB.NET

```
'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a rectangle field and add the properties
Dim rectangle As New PdfXfaRectangleField("rect1", New SizeF(100, 50))
'Set the fill color
rectangle.Border.FillColor = New PdfXfaSolidBrush(Color.Red)
```

```
'Add the rectangle field to the XFA form
mainForm.Fields.Add(rectangle)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()
```

UWP

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a rectangle field and add the properties
PdfXfaRectangleField rectangle = new PdfXfaRectangleField("rect1", new
SizeF(100, 50));
//Set the fill color
rectangle.Border.FillColor = new
PdfXfaSolidBrush(Color.FromArgb(0, 255, 0, 0));
//Add the rectangle field to the XFA form
mainForm.Fields.Add(rectangle);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a rectangle field and add the properties
PdfXfaRectangleField rectangle = new PdfXfaRectangleField("rect1", new
SizeF(100, 50));
//Set the fill color
rectangle.Border.FillColor = new
PdfXfaSolidBrush(Color.FromArgb(0, 255, 0, 0));
//Add the rectangle field to the XFA form
mainForm.Fields.Add(rectangle);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
```

```
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a rectangle field and add the properties
PdfXfaRectangleField rectangle = new PdfXfaRectangleField("rect1", new
SizeF(100, 50));
//Set the fill color
rectangle.Border.FillColor = new
PdfXfaSolidBrush(Syncfusion.Drawing.Color.FromArgb(0, 255, 0, 0));
//Add the rectangle field to the XFA form
mainForm.Fields.Add(rectangle);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

The below code snippet illustrates how to add the rectangle field to an existing PDF document.

C#

```

//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new
PdfLoadedXfaDocument("XfaForm.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a rectangle field and add the properties
PdfXfaRectangleField rectangle = new PdfXfaRectangleField("rect1", new
SizeF(100, 50));
//Set the fill color
rectangle.Border.FillColor = new PdfXfaSolidBrush(Color.Red);
//Add the rectangle to the existing XFA form.
loadedForm.Fields.Add(rectangle);
//Save the document
loadedDocument.Save("XfaForm.pdf");
//Close the document
loadedDocument.Close();

```

VB.NET

```

'Load the existing XFA document
Dim loadedDocument As New PdfLoadedXfaDocument("XfaForm.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Create a rectangle field and add the properties
Dim rectangle As New PdfXfaRectangleField("rect1", New SizeF(100, 50))
'Set the fill color
rectangle.Border.FillColor = New PdfXfaSolidBrush(Color.Red)
'Add the rectangle to the existing XFA form
loadedForm.Fields.Add(rectangle)
'Save the document
loadedDocument.Save("XfaForm.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.XfaForm.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a rectangle field and add the properties
PdfXfaRectangleField rectangle = new PdfXfaRectangleField("rect1", new
SizeF(100, 50));
//Set the fill color
rectangle.Border.FillColor = new
PdfXfaSolidBrush(Color.FromArgb(0, 255, 0, 0));
//Add the rectangle to the existing XFA form
loadedForm.Fields.Add(rectangle);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);

```

```
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a rectangle field and add the properties
PdfXfaRectangleField rectangle = new PdfXfaRectangleField("rect1", new
SizeF(100, 50));
//Set the fill color
rectangle.Border.FillColor = new
PdfXfaSolidBrush(Color.FromArgb(0, 255, 0, 0));
//Add the rectangle to the existing XFA form
loadedForm.Fields.Add(rectangle);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a rectangle field and add the properties
PdfXfaRectangleField rectangle = new PdfXfaRectangleField("rect1", new
SizeF(100, 50));
//Set the fill color
rectangle.Border.FillColor = new
PdfXfaSolidBrush(Syncfusion.Drawing.Color.FromArgb(0, 255, 0, 0));
//Add the rectangle to the existing XFA form
loadedForm.Fields.Add(rectangle);
MemoryStream stream = new MemoryStream();
```

```

loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Adding the XFA circle field

The below code snippet illustrates how to add the circle field to a new PDF document using [PdfXfaCircleField](#) class.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a circle field and add the properties
PdfXfaCircleField circle = new PdfXfaCircleField("circle", new SizeF(100,
100));
//Set the fill color
circle.Border.FillColor = new PdfXfaSolidBrush(Color.Red);
//Add the text element to the XFA form
mainForm.Fields.Add(circle);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form

```

```

Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a circle field and add the properties
Dim circle As New PdfXfaCircleField("circle", New SizeF(100, 100))
'Set the fill color
circle.Border.FillColor = New PdfXfaSolidBrush(Color.Red)
'Add the text element to the XFA form
mainForm.Fields.Add(circle)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a circle field and add the properties
PdfXfaCircleField circle = new PdfXfaCircleField("circle", new SizeF(100,
100));
//Set the fill color
circle.Border.FillColor = new PdfXfaSolidBrush(Color.FromArgb(0, 255, 0, 0));
//Add the text element to the XFA form
mainForm.Fields.Add(circle);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a circle field and add the properties
PdfXfaCircleField circle = new PdfXfaCircleField("circle", new SizeF(100,
100));
//Set the fill color
circle.Border.FillColor = new PdfXfaSolidBrush(Color.FromArgb(0, 255, 0, 0));
//Add the text element to the XFA form

```

```

mainForm.Fields.Add(circle);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a circle field and add the properties
PdfXfaCircleField circle = new PdfXfaCircleField("circle", new SizeF(100,
100));
//Set the fill color
circle.Border.FillColor = new
PdfXfaSolidBrush(Syncfusion.Drawing.Color.FromArgb(0,255,0,0));
//Add the text element to the XFA form
mainForm.Fields.Add(circle);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

```
}

```

The below code snippet illustrates how to add the circle field to an existing PDF document.

C#

```
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new
PdfLoadedXfaDocument("XfaForm.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a circle field and add the properties
PdfXfaCircleField circle = new PdfXfaCircleField("circle", new SizeF(100,
100));
//Set the fill color
circle.Border.FillColor = new PdfXfaSolidBrush(Color.Red);
//Add the circle to the existing XFA form
loadedForm.Fields.Add(circle);
//Save the document
loadedDocument.Save("XfaForm.pdf");
//Close the document
loadedDocument.Close();
```

VB.NET

```
'Load the existing XFA document
Dim loadedDocument As New PdfLoadedXfaDocument("XfaForm.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Create a circle field and add the properties
Dim circle As New PdfXfaCircleField("circle", New SizeF(100, 100))
'Set the fill color
circle.Border.FillColor = New PdfXfaSolidBrush(Color.Red)
'Add the circle to the existing XFA form
loadedForm.Fields.Add(circle)
'Save the document
loadedDocument.Save("XfaForm.pdf")
'Close the document
loadedDocument.Close()
```

UWP

```
//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.XfaForm.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a circle field and add the properties
PdfXfaCircleField circle = new PdfXfaCircleField("circle", new SizeF(100,
100));
//Set the fill color
circle.Border.FillColor = new PdfXfaSolidBrush(Color.FromArgb(0, 255, 0, 0));
```

```
//Add the circle to the existing XFA form
loadedForm.Fields.Add(circle);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a circle field and add the properties
PdfXfaCircleField circle = new PdfXfaCircleField("circle", new SizeF(100,
100));
//Set the fill color
circle.Border.FillColor = new PdfXfaSolidBrush(Color.FromArgb(0, 255, 0, 0));
//Add the circle to the existing XFA form
loadedForm.Fields.Add(circle);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a circle field and add the properties
PdfXfaCircleField circle = new PdfXfaCircleField("circle", new SizeF(100,
100));
//Set the fill color
circle.Border.FillColor = new
PdfXfaSolidBrush(Syncfusion.Drawing.Color.FromArgb(0, 255, 0, 0));
```

```

//Add the circle to the existing XFA form
loadedForm.Fields.Add(circle);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Adding the XFA line

The below code snippet illustrates how to add a line to a new PDF document using [PdfXfaLine](#) class.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a line and add the properties
PdfXfaLine line = new PdfXfaLine(new PointF(0, 0), new PointF(200, 0), 3);
//Set the line color
line.Color = new PdfColor(Color.Red);
//Add the text line to the XFA form
mainForm.Fields.Add(line);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form

```



```

Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a line and add the properties
Dim line As New PdfXfaLine(New PointF(0, 0), New PointF(200, 0), 3)
'Set the line color
line.Color = New PdfColor(Color.Red)
'Add the text line to the XFA form
mainForm.Fields.Add(line)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a line and add the properties
PdfXfaLine line = new PdfXfaLine(new PointF(0, 0), new PointF(200, 0), 3);
//Set the line color
line.Color = new PdfColor(Color.FromArgb(0, 255, 0, 0));
//Add the text line to the XFA form
mainForm.Fields.Add(line);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a line and add the properties
PdfXfaLine line = new PdfXfaLine(new PointF(0, 0), new PointF(200, 0), 3);
//Set the line color
line.Color = new PdfColor(Color.FromArgb(0, 255, 0, 0));
//Add the text line to the XFA form
mainForm.Fields.Add(line);
//Add the XFA form to the document

```

```

document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Create a line and add the properties
PdfXfaLine line = new PdfXfaLine(new PointF(0, 0), new PointF(200, 0), 3);
//Set the line color
line.Color = new PdfColor(Syncfusion.Drawing.Color.FromArgb(0, 255, 0, 0));
//Add the text line to the XFA form
mainForm.Fields.Add(line);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

The below code snippet illustrates how to add a line to an existing PDF document.

C#

```

//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new
PdfLoadedXfaDocument("XfaForm.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a line and add the properties
PdfXfaLine line = new PdfXfaLine(new PointF(0, 0), new PointF(200, 0), 3);
//Set the line color
line.Color = new PdfColor(Color.Red);
//Add the line to the existing XFA form
loadedForm.Fields.Add(line);
//Save the document
loadedDocument.Save("XfaForm.pdf");
//Close the document
loadedDocument.Close();

```

VB.NET

```

'Load the existing XFA document
Dim loadedDocument As New PdfLoadedXfaDocument("XfaForm.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Create a line and add the properties
Dim line As New PdfXfaLine(New PointF(0, 0), New PointF(200, 0), 3)
'Set the line color
line.Color = New PdfColor(Color.Red)
'Add the line to the existing XFA form
loadedForm.Fields.Add(line)
'Save the document
loadedDocument.Save("XfaForm.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.XfaForm.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a line and add the properties
PdfXfaLine line = new PdfXfaLine(new PointF(0, 0), new PointF(200, 0), 3);
//Set the line color
line.Color = new PdfColor(Color.FromArgb(0, 255, 0, 0));
//Add the line to the existing XFA form
loadedForm.Fields.Add(line);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();

```

```
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a line and add the properties
PdfXfaLine line = new PdfXfaLine(new PointF(0, 0), new PointF(200, 0), 3);
//Set the line color
line.Color = new PdfColor(Color.FromArgb(0,255,0,0));
//Add the line to the existing XFA form
loadedForm.Fields.Add(line);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a line and add the properties
PdfXfaLine line = new PdfXfaLine(new PointF(0, 0), new PointF(200, 0), 3);
//Set the line color
line.Color = new PdfColor(Syncfusion.Drawing.Color.FromArgb(0,255,0,0));
//Add the line to the existing XFA form
loadedForm.Fields.Add(line);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Adding the XFA image

The below code snippet illustrates how to add an image to a new PDF document using [PdfXfaImage](#) class.

C#

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width);
//Create a image and add the properties
PdfXfaImage image = new PdfXfaImage("image1", "image.jpg");
//Add the image to the XFA form
mainForm.Fields.Add(image);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();
```

VB.NET

```
'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add a new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form
Dim mainForm As New
PdfXfaForm("subform1", xfaPage, xfaPage.GetClientSize().Width)
'Create a image and add the properties
Dim image As New PdfXfaImage("image1", "image.jpg")
'Add the image to the XFA form
mainForm.Fields.Add(image)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
```

```
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()
```

UWP

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Load the image as stream
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.image.jpg");
//Create a image and add the properties
PdfXfaImage image = new PdfXfaImage("image1", imageStream);
//Add the image to the XFA form
mainForm.Fields.Add(image);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Load the image as stream
MemoryStream imageStream = new
MemoryStream(File.ReadAllBytes(imageFileName));
//Create a image and add the properties
PdfXfaImage image = new PdfXfaImage("image1", imageStream);
//Add the image to the XFA form
mainForm.Fields.Add(image);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
```

```

string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add a new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form
PdfXfaForm mainForm = new PdfXfaForm("subform1", xfaPage,
xfaPage.GetClientSize().Width);
//Load the image as stream
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
image.jpg");
//Create a image and add the properties
PdfXfaImage image = new PdfXfaImage("image1", imageStream);
//Add the image to the XFA form
mainForm.Fields.Add(image);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

The below code snippet illustrates how to add an image to an existing PDF document.

C#

```

//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new
PdfLoadedXfaDocument("XfaForm1.pdf");
//Load the existing XFA form

```

```

PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Create a image and add the properties
PdfXfaImage image = new PdfXfaImage("imgage1", "image.jpg");
//Add the image to the existing XFA form
loadedForm.Fields.Add(image);
//Save the document
loadedDocument.Save("XfaForm.pdf");
//Close the document
loadedDocument.Close();

```

VB.NET

```

'Load the existing XFA document
Dim loadedDocument As New PdfLoadedXfaDocument("XfaForm1.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Create a image and add the properties
Dim image As New PdfXfaImage("imgage1", "image.jpg")
'Add the image to the existing XFA form
loadedForm.Fields.Add(image)
'Save the document
loadedDocument.Save("XfaForm.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.XfaForm.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Load the image as stream
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.image.jpg");
//Create a image and add the properties
PdfXfaImage image = new PdfXfaImage("imgage1", imageStream);
//Add the image to the existing XFA form
loadedForm.Fields.Add(image);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Load the PDF document

```



```

FileStream docStream = new FileStream(fileName, FileMode.Open,
    FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Load the image as stream
MemoryStream imageStream = new
    MemoryStream(File.ReadAllBytes(imageFileName));
//Create a image and add the properties
PdfXfaImage image = new PdfXfaImage("imgage1", imageStream);
//Add the image to the existing XFA form
loadedForm.Fields.Add(image);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the PDF document
Stream docStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Load the image as stream
Stream imageStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
image.jpg");
//Create a image and add the properties
PdfXfaImage image = new PdfXfaImage("imgage1", imageStream);
//Add the image to the existing XFA form
loadedForm.Fields.Add(image);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Working with XFA form flow directions

There are two types of flow directions available in XFA forms.

1. Horizontal
2. Vertical

Horizontal flow direction

You can set the flow direction to an XFA form while creating, using [PdfXfaFlowDirection](#) Enum. The below sample illustrate how to set the horizontal flow direction in XFA forms.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
PdfXfaTextBoxField textBoxField1 = new PdfXfaTextBoxField("LastName", new
SizeF(200, 20));
//Set the caption text
textBoxField1.Caption.Text = "Last Name";
mainForm.Fields.Add(textBoxField1);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form with horizontal flow direction
Dim mainForm As New PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width)
'Create a textbox field and add the properties
Dim textBoxField As New PdfXfaTextBoxField("FirstName", New SizeF(200, 20))
'Set the caption text
textBoxField.Caption.Text = "First Name"
'Set the tool tip
textBoxField.ToolTip = "First Name"
'Add the field to the XFA form
mainForm.Fields.Add(textBoxField)
Dim textBoxField1 As New PdfXfaTextBoxField("LastName", New SizeF(200, 20))
'Set the caption text
textBoxField1.Caption.Text = "Last Name"
mainForm.Fields.Add(textBoxField1)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
PdfXfaTextBoxField textBoxField1 = new PdfXfaTextBoxField("LastName", new
SizeF(200, 20));
//Set the caption text
textBoxField1.Caption.Text = "Last Name";
mainForm.Fields.Add(textBoxField1);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();

```

```
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
PdfXfaTextBoxField textBoxField1 = new PdfXfaTextBoxField("LastName", new
SizeF(200, 20));
//Set the caption text
textBoxField1.Caption.Text = "Last Name";
mainForm.Fields.Add(textBoxField1);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
```

```

textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
PdfXfaTextBoxField textBoxField1 = new PdfXfaTextBoxField("LastName", new
SizeF(200, 20));
//Set the caption text
textBoxField1.Caption.Text = "Last Name";
mainForm.Fields.Add(textBoxField1);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Vertical flow direction

You can set the flow direction to an XFA form while creating, using [PdfXfaFlowDirection](#) Enum. The below sample illustrate how to set the vertical flow direction in XFA forms.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with vertical flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage, PdfXfaFlowDirection.Vertical,
xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);

```

```

PdfXfaTextBoxField textBoxField1 = new PdfXfaTextBoxField("LastName", new
SizeF(200, 20));
//Set the caption text
textBoxField1.Caption.Text = "Last Name";
mainForm.Fields.Add(textBoxField1);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form with vertical flow direction
Dim mainForm As New PdfXfaForm(xfaPage, PdfXfaFlowDirection.Vertical,
xfaPage.ClientSize().Width)
'Create a textbox field and add the properties
Dim textBoxField As New PdfXfaTextBoxField("FirstName", New SizeF(200, 20))
'Set the caption text
textBoxField.Caption.Text = "First Name"
'Set the tool tip
textBoxField.ToolTip = "First Name"
'Add the field to the XFA form
mainForm.Fields.Add(textBoxField)
Dim textBoxField1 As New PdfXfaTextBoxField("LastName", New SizeF(200, 20))
'Set the caption text
textBoxField1.Caption.Text = "Last Name"
mainForm.Fields.Add(textBoxField1)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with vertical flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage, PdfXfaFlowDirection.Vertical,
xfaPage.ClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";

```

```
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
PdfXfaTextBoxField textBoxField1 = new PdfXfaTextBoxField("LastName", new
SizeF(200, 20));
//Set the caption text
textBoxField1.Caption.Text = "Last Name";
mainForm.Fields.Add(textBoxField1);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with vertical flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage, PdfXfaFlowDirection.Vertical,
xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
PdfXfaTextBoxField textBoxField1 = new PdfXfaTextBoxField("LastName", new
SizeF(200, 20));
//Set the caption text
textBoxField1.Caption.Text = "Last Name";
mainForm.Fields.Add(textBoxField1);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with vertical flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage, PdfXfaFlowDirection.Vertical,
xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
PdfXfaTextBoxField textBoxField1 = new PdfXfaTextBoxField("LastName", new
SizeF(200, 20));
//Set the caption text
textBoxField1.Caption.Text = "Last Name";
mainForm.Fields.Add(textBoxField1);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Rotating XFA form fields

You can rotate a form field in XFA document, using [PdfXfaRotateAngle](#) Enum. The following code snippet explains this.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page

```



```

PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Set the fields rotation
textBoxField.Rotate = PdfXfaRotateAngle.RotateAngle90;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form with horizontal flow direction
Dim mainForm As New PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width)
'Create a textbox field and add the properties
Dim textBoxField As New PdfXfaTextBoxField("FirstName", New SizeF(200, 20))
'Set the caption text
textBoxField.Caption.Text = "First Name"
'Set the tool tip
textBoxField.ToolTip = "First Name"
'Set the fields rotation
textBoxField.Rotate = PdfXfaRotateAngle.RotateAngle90
'Add the field to the XFA form
mainForm.Fields.Add(textBoxField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction

```

```

PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Set the fields rotation
textBoxField.Rotate = PdfXfaRotateAngle.RotateAngle90;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Set the fields rotation
textBoxField.Rotate = PdfXfaRotateAngle.RotateAngle90;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";

```

```
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a textbox field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("FirstName", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the tool tip
textBoxField.ToolTip = "First Name";
//Set the fields rotation
textBoxField.Rotate = PdfXfaRotateAngle.RotateAngle90;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Validating the date time field

You can validate the date time fields of the input text by enabling the [RequireValidation](#) property of [PdfXfaDateTimeField](#). The following code snippet illustrates this.

C#

```
//Create a new PDF XFA document
```

```

PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a date time field and add the properties
PdfXfaDateTimeField dateTimeField = new PdfXfaDateTimeField("dateTimeField",
new SizeF(200, 20));
//Set the caption text
dateTimeField.Caption.Text = "Date of Birth";
//Set the date pattern
dateTimeField.DatePattern = PdfXfaDatePattern.DDMMYY;
//Enable the validation
dateTimeField.RequireValidation = true;
//Add the field to the XFA form
mainForm.Fields.Add(dateTimeField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form with horizontal flow direction
Dim mainForm As New PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width)
'Create a date time field and add the properties
Dim dateTimeField As New PdfXfaDateTimeField("dateTimeField", New SizeF(200,
20))
'Set the caption text
dateTimeField.Caption.Text = "Date of Birth"
'Set the date pattern
dateTimeField.DatePattern = PdfXfaDatePattern.DDMMYY
'Enable the validation
dateTimeField.RequireValidation = True
'Add the field to the XFA form
mainForm.Fields.Add(dateTimeField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page

```

```

PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a date time field and add the properties
PdfXfaDateTimeField dateTimeField = new PdfXfaDateTimeField("dateTimeField",
new SizeF(200, 20));
//Set the caption text
dateTimeField.Caption.Text = "Date of Birth";
//Set the date pattern
dateTimeField.DatePattern = PdfXfaDatePattern.DDMMYY;
//Enable the validation
dateTimeField.RequireValidation = true;
//Add the field to the XFA form
mainForm.Fields.Add(dateTimeField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a date time field and add the properties
PdfXfaDateTimeField dateTimeField = new PdfXfaDateTimeField("dateTimeField",
new SizeF(200, 20));
//Set the caption text
dateTimeField.Caption.Text = "Date of Birth";
//Set the date pattern
dateTimeField.DatePattern = PdfXfaDatePattern.DDMMYY;
//Enable the validation
dateTimeField.RequireValidation = true;
//Add the field to the XFA form
mainForm.Fields.Add(dateTimeField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name

```

```
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a date time field and add the properties
PdfXfaDateTimeField dateTimeField = new PdfXfaDateTimeField("dateTimeField",
new SizeF(200, 20));
//Set the caption text
dateTimeField.Caption.Text = "Date of Birth";
//Set the date pattern
dateTimeField.DatePattern = PdfXfaDatePattern.DDMMMYYYY;
//Enable the validation
dateTimeField.RequireValidation = true;
//Add the field to the XFA form
mainForm.Fields.Add(dateTimeField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Customizing the numeric field value

You can customize the numeric field input value to the specific pattern through [PatternString](#) property of [PdfXfaNumericField](#) class. The following code snippet explains this.

Please refer the below link for numeric pattern format in detail,

http://help.adobe.com/en_US/lifecycle/10.0/DesignerHelp/WS92d06802c76abadb-56c02439129b8b00ebc-7ecf.html

C#

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a numeric field and add the properties
PdfXfaNumericField numericField = new PdfXfaNumericField("numericField", new
SizeF(200, 20));
//Set the caption text
numericField.Caption.Text = "Numeric Field";
//Set the pattern string
numericField.PatternString = "zzzzzzzzz9";
//Add the field to the XFA form
mainForm.Fields.Add(numericField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();
```

VB.NET

```
'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form with horizontal flow direction
Dim mainForm As New PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width)
'Create a numeric field and add the properties
Dim numericField As New PdfXfaNumericField("numericField", New SizeF(200,
20))
'Set the caption text
numericField.Caption.Text = "Numeric Field"
'Set the pattern string
numericField.PatternString = "zzzzzzzzz9"
'Add the field to the XFA form
mainForm.Fields.Add(numericField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()
```

UWP

```
//Create a new PDF XFA document
```

```

PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a numeric field and add the properties
PdfXfaNumericField numericField = new PdfXfaNumericField("numericField", new
SizeF(200, 20));
//Set the caption text
numericField.Caption.Text = "Numeric Field";
//Set the pattern string
numericField.PatternString = "zzzzzzzz9";
//Add the field to the XFA form
mainForm.Fields.Add(numericField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a numeric field and add the properties
PdfXfaNumericField numericField = new PdfXfaNumericField("numericField", new
SizeF(200, 20));
//Set the caption text
numericField.Caption.Text = "Numeric Field";
//Set the pattern string
numericField.PatternString = "zzzzzzzz9";
//Add the field to the XFA form
mainForm.Fields.Add(numericField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";

```



```
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a numeric field and add the properties
PdfXfaNumericField numericField = new PdfXfaNumericField("numericField", new
SizeF(200, 20));
//Set the caption text
numericField.Caption.Text = "Numeric Field";
//Set the pattern string
numericField.PatternString = "zzzzzzzz9";
//Add the field to the XFA form
mainForm.Fields.Add(numericField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Adding nested sub forms

You can add the nested sub forms by using the following code snippet.

C#

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a font
```

```

PdfFont font = new PdfStandardFont(xfaPage, PdfFontFamily.Helvetica, 12,
PdfFontStyle.Bold);
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a text element
PdfXfaTextElement element = new PdfXfaTextElement("Main Form", font, 400,
20);
//Add the field to the XFA form
mainForm.Fields.Add(element);
//Create a form
PdfXfaForm form1 = new PdfXfaForm(PdfXfaFlowDirection.Vertical, 400);
PdfXfaTextElement element1 = new PdfXfaTextElement("First Form", font, 400,
20);
form1.Fields.Add(element1);
PdfXfaForm form2 = new PdfXfaForm(PdfXfaFlowDirection.Horizontal, 400);
PdfXfaTextElement element2 = new PdfXfaTextElement("Second Form",
font, 400, 20);
form2.Fields.Add(element2);
form1.Fields.Add(form2);
mainForm.Fields.Add(form1);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Static);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a font
Dim font As PdfFont = New PdfStandardFont(xfaPage, PdfFontFamily.Helvetica,
12, PdfFontStyle.Bold)
'Create a new PDF XFA form with horizontal flow direction
Dim mainForm As New PdfXfaForm(PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width)
'Create a text element
Dim element As New PdfXfaTextElement("Main Form", font, 400, 20)
'Add the field to the XFA form
mainForm.Fields.Add(element)
'Create a form
Dim form1 As New PdfXfaForm(PdfXfaFlowDirection.Vertical, 400)
Dim element1 As New PdfXfaTextElement("First Form", font, 400, 20)
form1.Fields.Add(element1)
Dim form2 As New PdfXfaForm(PdfXfaFlowDirection.Horizontal, 400)
Dim element2 As New PdfXfaTextElement("Second Form", font, 400, 20)
form2.Fields.Add(element2)
form1.Fields.Add(form2)
mainForm.Fields.Add(form1)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document

```

```
document.Save("XfaForm.pdf", PdfXfaType.Static)
'close the document
document.Close()
```

UWP

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Bold);
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a text element
PdfXfaTextElement element = new PdfXfaTextElement("Main Form", font, 400,
20);
//Add the field to the XFA form
mainForm.Fields.Add(element);
//Create a form
PdfXfaForm form1 = new PdfXfaForm(PdfXfaFlowDirection.Vertical, 400);
PdfXfaTextElement element1 = new PdfXfaTextElement("First Form", font, 400,
20);
form1.Fields.Add(element1);
PdfXfaForm form2 = new PdfXfaForm(PdfXfaFlowDirection.Horizontal, 400);
PdfXfaTextElement element2 = new PdfXfaTextElement("Second Form", font, 400,
20);
form2.Fields.Add(element2);
form1.Fields.Add(form2);
mainForm.Fields.Add(form1);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Bold);
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a text element
```

```

PdfXfaTextElement element = new PdfXfaTextElement("Main Form", font, 400,
20);
//Add the field to the XFA form
mainForm.Fields.Add(element);
//Create a form
PdfXfaForm form1 = new PdfXfaForm(PdfXfaFlowDirection.Vertical, 400);
PdfXfaTextElement element1 = new PdfXfaTextElement("First Form", font, 400,
20);
form1.Fields.Add(element1);
PdfXfaForm form2 = new PdfXfaForm(PdfXfaFlowDirection.Horizontal, 400);
PdfXfaTextElement element2 = new PdfXfaTextElement("Second Form", font, 400,
20);
form2.Fields.Add(element2);
form1.Fields.Add(form2);
mainForm.Fields.Add(form1);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Bold);
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a text element
PdfXfaTextElement element = new PdfXfaTextElement("Main Form", font, 400,
20);
//Add the field to the XFA form
mainForm.Fields.Add(element);
//Create a form
PdfXfaForm form1 = new PdfXfaForm(PdfXfaFlowDirection.Vertical, 400);
PdfXfaTextElement element1 = new PdfXfaTextElement("First Form", font, 400,
20);
form1.Fields.Add(element1);
PdfXfaForm form2 = new PdfXfaForm(PdfXfaFlowDirection.Horizontal, 400);
PdfXfaTextElement element2 = new PdfXfaTextElement("Second Form", font, 400,
20);

```

```

form2.Fields.Add(element2);
form1.Fields.Add(form2);
mainForm.Fields.Add(form1);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Formatting XFA form fields

Working with alignments

XFA support both the horizontal and vertical alignments.

Horizontal alignment

You can add the horizontal alignments in XFA form fields through [HorizontalAlignment](#) property by using [PdfXfaHorizontalAlignment](#) Enum. The below code snippet illustrates this.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the horizontal alignment
textBoxField.HorizontalAlignment = PdfXfaHorizontalAlignment.Right;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document

```

```
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();
```

VB.NET

```
'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form with horizontal flow direction
Dim mainForm As New PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width)
'Create a text box field and add the properties
Dim textBoxField As New PdfXfaTextBoxField("textBoxField", New SizeF(200,
20))
'Set the caption text
textBoxField.Caption.Text = "Fist Name"
'Set the horizontal alignment
textBoxField.HorizontalAlignment = PdfXfaHorizontalAlignment.Right
'Add the field to the XFA form
mainForm.Fields.Add(textBoxField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()
```

UWP

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the horizontal alignment
textBoxField.HorizontalAlignment = PdfXfaHorizontalAlignment.Right;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the horizontal alignment
textBoxField.HorizontalAlignment = PdfXfaHorizontalAlignment.Right;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the horizontal alignment
textBoxField.HorizontalAlignment = PdfXfaHorizontalAlignment.Right;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
```

```

document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Vertical alignment

You can add the vertical alignments in XFA form fields through [VerticalAlignment](#) property by using [PdfXfaVerticalAlignment](#) Enum. The below code snippet explains this.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(XfaPagePdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the vertical alignment
textBoxField.VerticalAlignment = PdfXfaVerticalAlignment.Bottom;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add new XFA page

```



```

Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form with horizontal flow direction
Dim mainForm As New PdfXfaForm(XfaPagePdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width)
'Create a text box field and add the properties
Dim textBoxField As New PdfXfaTextBoxField("textBoxField", New SizeF(200,
20))
'Set the caption text
textBoxField.Caption.Text = "Fist Name"
'Set the vertical alignment
textBoxField.VerticalAlignment = PdfXfaVerticalAlignment.Bottom
'Add the field to the XFA form
mainForm.Fields.Add(textBoxField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the vertical alignment
textBoxField.VerticalAlignment = PdfXfaVerticalAlignment.Bottom;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction

```

```

PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the vertical alignment
textBoxField.VerticalAlignment = PdfXfaVerticalAlignment.Bottom;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the vertical alignment
textBoxField.VerticalAlignment = PdfXfaVerticalAlignment.Bottom;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file

```

```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Adding margins to the XFA fields

You can add margin to the XFA form fields by using [Margins](#) property. The below code snippet explains this.

C#

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the margins
textBoxField.Margins.Left = 10;
textBoxField.Margins.Right = 5;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();
```

VB.NET

```
'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form with horizontal flow direction
Dim mainForm As New PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width)
'Create a text box field and add the properties
```

```

Dim textBoxField As New PdfXfaTextBoxField("textBoxField", New SizeF(200,
20))
'Set the caption text
textBoxField.Caption.Text = "Fist Name"
'Set the margins
textBoxField.Margins.Left = 10
textBoxField.Margins.Right = 5
'Add the field to the XFA form
mainForm.Fields.Add(textBoxField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the margins
textBoxField.Margins.Left = 10;
textBoxField.Margins.Right = 5;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a text box field and add the properties

```

```

PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the margins
textBoxField.Margins.Left = 10;
textBoxField.Margins.Right = 5;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.ClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the margins
textBoxField.Margins.Left = 10;
textBoxField.Margins.Right = 5;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Adding padding to the XFA fields

You can add padding to the XFA form fields by using the below code snippet.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the paddings
textBoxField.Padding.All = 2;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form with horizontal flow direction
Dim mainForm As New PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width)
'Create a text box field and add the properties
Dim textBoxField As New PdfXfaTextBoxField("textBoxField", New SizeF(200,
20))
'Set the caption text
textBoxField.Caption.Text = "Fist Name"
'Set the paddings
textBoxField.Padding.All = 2

```

```

'Add the field to the XFA form
mainForm.Fields.Add(textBoxField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.ClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the paddings
textBoxField.Padding.All = 2;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.ClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the paddings
textBoxField.Padding.All = 2;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document

```

```

document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the paddings
textBoxField.Padding.All = 2;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```


Adding border to the XFA fields

You can add the fields border by using the below code snippet.

C#

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the border color and width
textBoxField.Border.Color = new PdfColor (200,123,0);
textBoxField.Border.Width = 2;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();
```

VB.NET

```
'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form with horizontal flow direction
Dim mainForm As New PdfXfaForm(xfaPage, PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width)
'Create a text box field and add the properties
Dim textBoxField As New PdfXfaTextBoxField("textBoxField", New SizeF(200,
20))
'Set the caption text
textBoxField.Caption.Text = "Fist Name"
'Set the border color and width
textBoxField.Border.Color = New PdfColor(200,123,0)
textBoxField.Border.Width = 2
'Add the field to the XFA form
mainForm.Fields.Add(textBoxField)
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()
```

UWP

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the border color and width
textBoxField.Border.Color = new PdfColor(200, 123, 0);
textBoxField.Border.Width = 2;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "XfaForm.pdf");
```

ASP.NET CORE

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Set the border color and width
textBoxField.Border.Color = new PdfColor(200, 123, 0);
textBoxField.Border.Width = 2;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
```

```
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(xfaPage,
PdfXfaFlowDirection.Horizontal, xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "First Name";
//Set the border color and width
textBoxField.Border.Color = new PdfColor(200, 123, 0);
textBoxField.Border.Width = 2;
//Add the field to the XFA form
mainForm.Fields.Add(textBoxField);
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Filling form fields in an existing XFA document

Essential PDF allows you to fill the XFA form fields using [PdfLoadedXfaform](#) class.

Filling the XFA text box field

Please refer the below sample for filling the XFA text box field using [PdfLoadedXfaTextBoxField](#) class.

C#

```
//Load the existing PDF document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument("input.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded text box field
PdfLoadedXfaTextBoxField loadedTextBox = (loadedForm.Fields["subform1[0]"]
as PdfLoadedXfaForm).Fields["text[0]"] as PdfLoadedXfaTextBoxField;
//fill the text box
loadedTextBox.Text = "First Name";
//Save the document
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close();
```

VB.NET

```
'Load the existing PDF document
Dim loadedDocument As New PdfLoadedXfaDocument("input.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Get the loaded text box field
Dim loadedTextBox As PdfLoadedXfaTextBoxField =
TryCast((TryCast(loadedForm.Fields("subform1[0]"),
PdfLoadedXfaForm).Fields("text[0]"), PdfLoadedXfaTextBoxField)
'fill the text box
loadedTextBox.Text = "First Name"
'Save the document
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close()
```

UWP

```
//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded text box field
PdfLoadedXfaTextBoxField loadedTextBox = (loadedForm.Fields["subform1[0]"]
as PdfLoadedXfaForm).Fields["text[0]"] as PdfLoadedXfaTextBoxField;
//fill the text box
loadedTextBox.Text = "First Name";
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
```

```
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded text box field
PdfLoadedXfaTextBoxField loadedTextBox = (loadedForm.Fields["subform1[0]"]
as PdfLoadedXfaForm).Fields["text[0]"] as PdfLoadedXfaTextBoxField;
//fill the text box
loadedTextBox.Text = "First Name";
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded text box field
PdfLoadedXfaTextBoxField loadedTextBox = (loadedForm.Fields["subform1[0]"]
as PdfLoadedXfaForm).Fields["text[0]"] as PdfLoadedXfaTextBoxField;
//fill the text box
loadedTextBox.Text = "First Name";
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Filling the XFA numeric field

You can fill the XFA numeric field by using [PdfLoadedXfaNumericField](#) class. The below code snippet illustrates this.

C#

```
//Load the existing PDF document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument("input.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded numeric field
PdfLoadedXfaNumericField loadedNumericField =
(loaderForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["numericField[0]"] as PdfLoadedXfaNumericField;
//fill the numeric field
loadedNumericField.NumericValue = 945322;
//Save the document
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close();
```

VB.NET

```
'Load the existing PDF document
Dim loadedDocument As New PdfLoadedXfaDocument("input.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Get the loaded numeric field
Dim loadedNumericField As PdfLoadedXfaNumericField =
TryCast((TryCast(loadedForm.Fields("subform1[0]"),
PdfLoadedXfaForm).Fields("numericField[0]"), PdfLoadedXfaNumericField)
'fill the numeric field
loadedNumericField.NumericValue = 945322
'Save the document
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close()
```

UWP

```

//Load the PDF document as stream
Stream docStream =
    typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded numeric field
PdfLoadedXfaNumericField loadedNumericField =
    (loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["numericField[0]"] as PdfLoadedXfaNumericField;
//fill the numeric field
loadedNumericField.NumericValue = 945322;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded numeric field
PdfLoadedXfaNumericField loadedNumericField =
    (loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["numericField[0]"] as PdfLoadedXfaNumericField;
//fill the numeric field
loadedNumericField.NumericValue = 945322;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document.
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the PDF document

```

```

Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded numeric field
PdfLoadedXfaNumericField loadedNumericField =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["numericField[0]"] as PdfLoadedXfaNumericField;
//fill the numeric field
loadedNumericField.NumericValue = 945322;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document.
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Filling the XFA combo box field

You can fill the XFA combo box field by using [PdfLoadedXfaComboBoxField](#) class. The below code snippet explains this.

C#

```

//Load the existing PDF document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument("input.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded combo box field
PdfLoadedXfaComboBoxField loadedComboBoxField =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["comboBoxField[0]"] as PdfLoadedXfaComboBoxField;
//Set the combo box selected index
loadedComboBoxField.SelectedIndex = 1;
//Save the document
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close();

```


VB.NET

```

'Load the existing PDF document
Dim loadedDocument As New PdfLoadedXfaDocument("input.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Get the loaded combo box field
Dim loadedComboBoxField As PdfLoadedXfaComboBoxField =
TryCast((TryCast(loadedForm.Fields("subform1[0]"),
PdfLoadedXfaForm)).Fields("comboBoxField[0]"), PdfLoadedXfaComboBoxField)
'Set the combo box selected index
loadedComboBoxField.SelectedIndex = 1
'Save the document
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded combo box field
PdfLoadedXfaComboBoxField loadedComboBoxField =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["comboBoxField[0]"] as PdfLoadedXfaComboBoxField;
//Set the combo box selected index
loadedComboBoxField.SelectedIndex = 1;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded combo box field.
PdfLoadedXfaComboBoxField loadedComboBoxField =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["comboBoxField[0]"] as PdfLoadedXfaComboBoxField;
//Set the combo box selected index
loadedComboBoxField.SelectedIndex = 1;

```

```

MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded combo box field.
PdfLoadedXfaComboBoxField loadedComboBoxField =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["comboBoxField[0]"] as PdfLoadedXfaComboBoxField;
//Set the combo box selected index
loadedComboBoxField.SelectedIndex = 1;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

You can fill and save hidden items in XFA combo box field by using the following code snippet.

C#

```

//Load the existing PDF document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument("Input.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded combo box field
PdfLoadedXfaComboBoxField loadedComboBoxField =
    (loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["comboBoxField[0]"] as PdfLoadedXfaComboBoxField;
//Set the combo box selected index using hidden items
loadedComboBoxField.SelectedValue = loadedComboBoxField.HiddenItems[0];
//Save the combo box field hidden values
List<string> hiddenValues = new List<string>();
hiddenValues = loadedComboBoxField.HiddenItems;
//Save the document
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close();

```

VB.NET

```

'Load the existing PDF document
Dim loadedDocument As New PdfLoadedXfaDocument("Input.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Get the loaded combo box field
Dim loadedComboBoxField As PdfLoadedXfaComboBoxField =
TryCast(TryCast(loadedForm.Fields("subform1[0]"),
PdfLoadedXfaForm).Fields("comboBoxField[0]"), PdfLoadedXfaComboBoxField)
'Set the combo box selected index using hidden items
loadedComboBoxField.SelectedValue = loadedComboBoxField.HiddenItems(0)
'Save the combo box field hidden values
Dim hiddenValues As New List(Of String)()
hiddenValues = loadedComboBoxField.HiddenItems
'Save the document
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded combo box field
PdfLoadedXfaComboBoxField loadedComboBoxField =
    (loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["comboBoxField[0]"] as PdfLoadedXfaComboBoxField;
//Set the combo box selected index using hidden items
loadedComboBoxField.SelectedValue = loadedComboBoxField.HiddenItems[0];
//Save the combo box field hidden values
List<string> hiddenValues = new List<string>();

```

```

hiddenValues = loadedComboBoxField.HiddenItems;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded combo box field
PdfLoadedXfaComboBoxField loadedComboBoxField =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["comboBoxField[0]"] as PdfLoadedXfaComboBoxField;
//Set the combo box selected index using hidden items
loadedComboBoxField.SelectedValue = loadedComboBoxField.HiddenItems[0];
//Save the combo box field hidden values
List<string> hiddenValues = new List<string>();
hiddenValues = loadedComboBoxField.HiddenItems;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded combo box field
PdfLoadedXfaComboBoxField loadedComboBoxField =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["comboBoxField[0]"] as PdfLoadedXfaComboBoxField;
//Set the combo box selected index using hidden items

```

```
loadedComboBoxField.SelectedValue = loadedComboBoxField.HiddenItems[0];
//Save the combo box field hidden values
List<string> hiddenValues = new List<string>();
hiddenValues = loadedComboBoxField.HiddenItems;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Filling the XFA list box field

You can fill the XFA list box field by using [PdfLoadedXfaListBoxField](#) class. The below code snippet illustrates this.

C#

```
//Load the existing PDF document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument("input.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded list box field
PdfLoadedXfaListBoxField loadedListBoxField =
(loaderForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["listBoxField[0]"] as PdfLoadedXfaListBoxField;
//Set the list box selected index
loadedListBoxField.SelectedIndex = 1;
//Save the document
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close();
```

VB.NET

```
'Load the existing PDF document
Dim loadedDocument As New PdfLoadedXfaDocument("input.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Get the loaded list box field
```

```

Dim loadedListBoxField As PdfLoadedXfaListBoxField =
TryCast((TryCast(loadedForm.Fields("subform1[0]"),
PdfLoadedXfaForm)).Fields("listBoxField[0]"), PdfLoadedXfaListBoxField)
'Set the list box selected index
loadedListBoxField.SelectedIndex = 1
'Save the document
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded list box field
PdfLoadedXfaListBoxField loadedListBoxField =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["listBoxField[0]"] as PdfLoadedXfaListBoxField;
//Set the list box selected index
loadedListBoxField.SelectedIndex = 1;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded list box field.
PdfLoadedXfaListBoxField loadedListBoxField =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["listBoxField[0]"] as PdfLoadedXfaListBoxField;
//Set the list box selected index
loadedListBoxField.SelectedIndex = 1;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file

```

```

string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded list box field.
PdfLoadedXfaListBoxField loadedListBoxField =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["listBoxField[0]"] as PdfLoadedXfaListBoxField;
//Set the list box selected index
loadedListBoxField.SelectedIndex = 1;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

You can also select the multiple values by using the below code snippet.

C#

```

//Load the existing PDF document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument("input.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded list box field

```

```

PdfLoadedXfaListBoxField loadedListBoxField =
    (loadedForm.Fields["subform1[0]"] as
    PdfLoadedXfaForm).Fields["listBoxField[0]"] as PdfLoadedXfaListBoxField;
//Set the list box selected items
loadedListBoxField.SelectedItems = new string[] { "English", "Spanish" };
//Save the document
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close();

```

VB.NET

```

'Load the existing PDF document
Dim loadedDocument As New PdfLoadedXfaDocument("input.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Get the loaded list box field
Dim loadedListBoxField As PdfLoadedXfaListBoxField =
    TryCast((TryCast(loadedForm.Fields("subform1[0]"),
    PdfLoadedXfaForm).Fields("listBoxField[0]"), PdfLoadedXfaListBoxField)
'Set the list box selected items
loadedListBoxField.SelectedItems = New String() { "English", "Spanish" }
'Save the document
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
    typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
    sets.Data.input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded list box field
PdfLoadedXfaListBoxField loadedListBoxField =
    (loadedForm.Fields["subform1[0]"] as
    PdfLoadedXfaForm).Fields["listBoxField[0]"] as PdfLoadedXfaListBoxField;
//Set the list box selected items
loadedListBoxField.SelectedItems = new string[] { "English", "Spanish" };
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document

```



```

FileStream docStream = new FileStream(fileName, FileMode.Open,
    FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded list box field
PdfLoadedXfaListBoxField loadedListBoxField =
    (loadedForm.Fields["subform1[0]"] as
    PdfLoadedXfaForm).Fields["listBoxField[0]"] as PdfLoadedXfaListBoxField;
//Set the list box selected items
loadedListBoxField.SelectedItems = new string[] { "English", "Spanish" };
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document.
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the PDF document
Stream docStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
    input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded list box field
PdfLoadedXfaListBoxField loadedListBoxField =
    (loadedForm.Fields["subform1[0]"] as
    PdfLoadedXfaForm).Fields["listBoxField[0]"] as PdfLoadedXfaListBoxField;
//Set the list box selected items
loadedListBoxField.SelectedItems = new string[] { "English", "Spanish" };
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document.
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
{

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Filling the XFA date time field

You can fill the XFA date time field by using [PdfLoadedXfaDateTimeField](#) class. The below code snippet explains this.

C#

```
//Load the existing PDF document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument("input.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded date time field
PdfLoadedXfaDateTimeField loadedDateTimeField =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["dateTimeField[0]"] as PdfLoadedXfaDateTimeField;
//Set the value
loadedDateTimeField.Value = DateTime.Now;
//Save the document
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close();
```

VB.NET

```
'Load the existing PDF document
Dim loadedDocument As New PdfLoadedXfaDocument("input.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Get the loaded date time field
Dim loadedDateTimeField As PdfLoadedXfaDateTimeField =
TryCast((TryCast(loadedForm.Fields("subform1[0]"),
PdfLoadedXfaForm).Fields("dateTimeField[0]"), PdfLoadedXfaDateTimeField)
'Set the value
loadedDateTimeField.Value = Date.Now
'Save the document
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close()
```

UWP

```
//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
```

```

//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded date time field
PdfLoadedXfaDateTimeField loadedDateTimeField =
    (loadedForm.Fields["subform1[0]"] as
    PdfLoadedXfaForm).Fields["dateTimeField[0]"] as PdfLoadedXfaDateTimeField;
//Set the value
loadedDateTimeField.Value = DateTime.Now;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
    FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded date time field
PdfLoadedXfaDateTimeField loadedDateTimeField =
    (loadedForm.Fields["subform1[0]"] as
    PdfLoadedXfaForm).Fields["dateTimeField[0]"] as PdfLoadedXfaDateTimeField;
//Set the value
loadedDateTimeField.Value = DateTime.Now;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the PDF document
Stream docStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
    input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded date time field

```

```

PdfLoadedXfaDateTimeField loadedDateTimeField =
    (loadedForm.Fields["subform1[0]"] as
    PdfLoadedXfaForm).Fields["dateTimeField[0]"] as PdfLoadedXfaDateTimeField;
//Set the value
loadedDateTimeField.Value = DateTime.Now;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
    "application/pdf", fdfStream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
    "application/pdf", fdfStream);
}

```

Filling the XFA check box field

You can fill the XFA check box field by using [PdfLoadedXfaCheckBoxField](#) class. The below code snippet illustrates this.

C#

```

//Load the existing PDF document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument("input.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded check box field
PdfLoadedXfaCheckBoxField loadedCheckBox = (loadedForm.Fields["subform1[0]"]
as PdfLoadedXfaForm).Fields["checkBox[0]"] as PdfLoadedXfaCheckBoxField;
//Check the check box
loadedCheckBox.IsChecked = true;
//Save the document
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close();

```

VB.NET

```

'Load the existing PDF document
Dim loadedDocument As New PdfLoadedXfaDocument("input.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Get the loaded check box field

```

```

Dim loadedCheckBox As PdfLoadedXfaCheckBoxField =
TryCast((TryCast(loadedForm.Fields("subform1[0]"),
PdfLoadedXfaForm)).Fields("checkBox[0]"), PdfLoadedXfaCheckBoxField)
'Check the check box
loadedCheckBox.IsChecked = True
'Save the document
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded check box field
PdfLoadedXfaCheckBoxField loadedCheckBox = (loadedForm.Fields["subform1[0]"]
as PdfLoadedXfaForm).Fields["checkBox[0]"] as PdfLoadedXfaCheckBoxField;
//Check the check box
loadedCheckBox.IsChecked = true;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded check box field
PdfLoadedXfaCheckBoxField loadedCheckBox = (loadedForm.Fields["subform1[0]"]
as PdfLoadedXfaForm).Fields["checkBox[0]"] as PdfLoadedXfaCheckBoxField;
//Check the check box
loadedCheckBox.IsChecked = true;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name

```

```
string fileName = "XfaForm.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the loaded check box field
PdfLoadedXfaCheckBoxField loadedCheckBox = (loadedForm.Fields["subform1[0]"]
as PdfLoadedXfaForm).Fields["checkBox[0]"] as PdfLoadedXfaCheckBoxField;
//Check the check box
loadedCheckBox.IsChecked = true;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Filling the XFA radio button field

You can fill the XFA radio button field by using [PdfLoadedXfaRadioButtonField](#) class. The below code snippet explains this.

C#

```
//Load the existing PDF document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument("input.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the radio button group
PdfLoadedXfaRadioButtonGroup loadedRadioButtonGroup =
(loadedForm.Fields["subform1[0]"] as
```

```

PdfLoadedXfaForm).Fields["radioButtonGroup[0]"] as
PdfLoadedXfaRadioButtonGroup;
//Get the radio button field
PdfLoadedXfaRadioButtonField loadedRadioButtonField =
loadedRadioButtonGroup.Fields[0] as PdfLoadedXfaRadioButtonField;
//Check the radio button
loadedRadioButtonField.IsChecked = true;
//Save the document
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close();

```

VB.NET

```

'Load the existing PDF document
Dim loadedDocument As New PdfLoadedXfaDocument("input.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Get the radio button group
Dim loadedRadioButtonGroup As PdfLoadedXfaRadioButtonGroup =
TryCast((TryCast(loadedForm.Fields("subform1[0]"),
PdfLoadedXfaForm).Fields("radioButtonGroup[0]"),
PdfLoadedXfaRadioButtonGroup)
'Get the radio button field
Dim loadedRadioButtonField As PdfLoadedXfaRadioButtonField =
TryCast(loadedRadioButtonGroup.Fields(0), PdfLoadedXfaRadioButtonField)
'Check the radio button
loadedRadioButtonField.IsChecked = True
'Save the document
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the radio button group
PdfLoadedXfaRadioButtonGroup loadedRadioButtonGroup =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["radioButtonGroup[0]"] as
PdfLoadedXfaRadioButtonGroup;
//Get the radio button field
PdfLoadedXfaRadioButtonField loadedRadioButtonField =
loadedRadioButtonGroup.Fields[0] as PdfLoadedXfaRadioButtonField;
//Check the radio button
loadedRadioButtonField.IsChecked = true;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document

```

```
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the radio button group
PdfLoadedXfaRadioButtonGroup loadedRadioButtonGroup =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["radioButtonGroup[0]"] as
PdfLoadedXfaRadioButtonGroup;
//Get the radio button field
PdfLoadedXfaRadioButtonField loadedRadioButtonField =
loadedRadioButtonGroup.Fields[0] as PdfLoadedXfaRadioButtonField;
//Check the radio button
loadedRadioButtonField.IsChecked = true;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the radio button group
PdfLoadedXfaRadioButtonGroup loadedRadioButtonGroup =
(loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["radioButtonGroup[0]"] as
PdfLoadedXfaRadioButtonGroup;
//Get the radio button field
PdfLoadedXfaRadioButtonField loadedRadioButtonField =
loadedRadioButtonGroup.Fields[0] as PdfLoadedXfaRadioButtonField;
```



```

//Check the radio button
loadedRadioButtonField.IsChecked = true;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Removing editing capability of form fields

The form field editing or filling capabilities can be removed by marking the form or field as read only. To prevent the user from changing the form field content, you can enable the [ReadOnly](#) property of [PdfXfaForm](#).

The below code snippet illustrates how to set the read only property to a new PDF document.

C#

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Add the text box to the XFA form
mainForm.Fields.Add(textBoxField);
//Set the form as read only
mainForm.ReadOnly = true;
//Add the XFA form to the document
document.XfaForm = mainForm;
//Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic);
//close the document
document.Close();

```

VB.NET

```

'Create a new PDF XFA document
Dim document As New PdfXfaDocument()
'Add new XFA page
Dim xfaPage As PdfXfaPage = document.Pages.Add()
'Create a new PDF XFA form with horizontal flow direction
Dim mainForm As New PdfXfaForm(PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width)
'Create a text box field and add the properties
Dim textBoxField As New PdfXfaTextBoxField("textBoxField", New SizeF(200,
20))
'Set the caption text
textBoxField.Caption.Text = "Fist Name"
'Add the text box to the XFA form
mainForm.Fields.Add(textBoxField)
'Set the form as read only
mainForm.ReadOnly = True
'Add the XFA form to the document
document.XfaForm = mainForm
'Save the document
document.Save("XfaForm.pdf", PdfXfaType.Dynamic)
'close the document
document.Close()

```

UWP

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Add the text box to the XFA form
mainForm.Fields.Add(textBoxField);
//Set the form as read only
mainForm.ReadOnly = true;
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//Close the document
document.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Create a new PDF XFA document

```

```

PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Add the text box to the XFA form
mainForm.Fields.Add(textBoxField);
//Set the form as read only
mainForm.ReadOnly = true;
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF XFA document
PdfXfaDocument document = new PdfXfaDocument();
//Add new XFA page
PdfXfaPage xfaPage = document.Pages.Add();
//Create a new PDF XFA form with horizontal flow direction
PdfXfaForm mainForm = new PdfXfaForm(PdfXfaFlowDirection.Horizontal,
xfaPage.GetClientSize().Width);
//Create a text box field and add the properties
PdfXfaTextBoxField textBoxField = new PdfXfaTextBoxField("textBoxField", new
SizeF(200, 20));
//Set the caption text
textBoxField.Caption.Text = "Fist Name";
//Add the text box to the XFA form
mainForm.Fields.Add(textBoxField);
//Set the form as read only
mainForm.ReadOnly = true;
//Add the XFA form to the document
document.XfaForm = mainForm;
MemoryStream stream = new MemoryStream();
document.Save(stream, PdfXfaType.Dynamic);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document

```

```
document.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

The below code snippet illustrates how to set the **ReadOnly** property to an existing PDF document.

C#

```
//Load the existing PDF document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument("input.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Set the form as read only
loadedForm.ReadOnly = true;
//Save the document
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close();
```

VB.NET

```
'Load the existing PDF document
Dim loadedDocument As New PdfLoadedXfaDocument("input.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Set the form as read only
loadedForm.ReadOnly = True
'Save the document
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close()
```

UWP

```
//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
```

```
//Set the form as read only
loadedForm.ReadOnly = true;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Set the form as read only
loadedForm.ReadOnly = true;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Set the form as read only
loadedForm.ReadOnly = true;
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Flattening XFA Form fields

The form field editing or filling capabilities can be removed by flattening the PDF document.

The Essential PDF flattens a form field by removing the existing form field and replacing it with graphical objects that will resemble the form field and cannot be edited. This can be achieved by enabling the [Flatten](#) property of [PdfLoadedXfaDocument](#).

The following code snippet illustrates how to flatten the XFA form field.

C#

```
//Load the existing document
PdfLoadedXfaDocument ldoc = new PdfLoadedXfaDocument("input.pdf");
//Set flatten
ldoc.Flatten = true;
//Save the document
ldoc.Save("output.pdf");
//Close the document
ldoc.Close();
```

VB.NET

```
'Load the existing document
Dim ldoc As New PdfLoadedXfaDocument("input.pdf")
'Set flatten
ldoc.Flatten = True
'Save the document
ldoc.Save("output.pdf")
'Close the document
ldoc.Close()
```

UWP

```
//PDF supports flattening XFA form fields only in Windows Forms, WPF,
//ASP.NET and ASP.NET MVC.
```

Removing the dynamic form fields from an existing XFA document

You can remove the dynamic XFA form fields by using [Remove](#) method of [PdfLoadedXfaFieldCollection](#) class. The below code snippet explained this.

C#

```

//Load the existing PDF document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument("input.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the existing field
PdfLoadedXfaTextBoxField loadedText = (loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["text[0]"] as PdfLoadedXfaTextBoxField;
//Remove the field
loadedForm.Fields.Remove(loadedText);
//Save the document
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close();

```

VB.NET

```

'Load the existing PDF document
Dim loadedDocument As New PdfLoadedXfaDocument("input.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Get the existing field
Dim loadedText As PdfLoadedXfaTextBoxField =
TryCast((TryCast(loadedForm.Fields("subform1[0]"),
PdfLoadedXfaForm).Fields("text[0]"), PdfLoadedXfaTextBoxField)
'Remove the field
loadedForm.Fields.Remove(loadedText)
'Save the document
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close()

```

UWP

```

//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the existing field
PdfLoadedXfaTextBoxField loadedText = (loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["text[0]"] as PdfLoadedXfaTextBoxField;
//Remove the field
loadedForm.Fields.Remove(loadedText);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the existing field
PdfLoadedXfaTextBoxField loadedText = (loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["text[0]"] as PdfLoadedXfaTextBoxField;
//Remove the field
loadedForm.Fields.Remove(loadedText);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the existing field
PdfLoadedXfaTextBoxField loadedText = (loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["text[0]"] as PdfLoadedXfaTextBoxField;
//Remove the field
loadedForm.Fields.Remove(loadedText);
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
```



```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Modifying the existing fields in XFA document

You can modify the existing dynamic XFA fields by using the below code snippet.

C#

```
//Load the existing PDF document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument("input.pdf");
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the existing field
PdfLoadedXfaTextBoxField loadedText = (loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["text[0]"] as PdfLoadedXfaTextBoxField;
//Set the width and height of the field
loadedText.Width = 200;
loadedText.Height = 30;
//Set the caption text
loadedText.Caption.Text = "Second Name";
//Save the document
loadedDocument.Save("output.pdf");
//Close the document
loadedDocument.Close();
```

VB.NET

```
'Load the existing PDF document
Dim loadedDocument As New PdfLoadedXfaDocument("input.pdf")
'Load the existing XFA form
Dim loadedForm As PdfLoadedXfaForm = loadedDocument.XfaForm
'Get the existing field
Dim loadedText As PdfLoadedXfaTextBoxField =
TryCast((TryCast(loadedForm.Fields("subform1[0]"),
PdfLoadedXfaForm).Fields("text[0]"), PdfLoadedXfaTextBoxField)
'Set the width and height of the field
loadedText.Width = 200
loadedText.Height = 30
'Set the caption text
loadedText.Caption.Text = "Second Name"
'Save the document
loadedDocument.Save("output.pdf")
'Close the document
loadedDocument.Close()
```

UWP

```
//Load the PDF document as stream
```

```

Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the existing field
PdfLoadedXfaTextBoxField loadedText = (loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["text[0]"] as PdfLoadedXfaTextBoxField;
//Set the width and height of the field
loadedText.Width = 200;
loadedText.Height = 30;
//Set the caption text
loadedText.Caption.Text = "Second Name";
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the existing field
PdfLoadedXfaTextBoxField loadedText = (loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["text[0]"] as PdfLoadedXfaTextBoxField;
//Set the width and height of the field
loadedText.Width = 200;
loadedText.Height = 30;
//Set the caption text
loadedText.Caption.Text = "Second Name";
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Load the existing XFA form
PdfLoadedXfaForm loadedForm = loadedDocument.XfaForm;
//Get the existing field
PdfLoadedXfaTextBoxField loadedText = (loadedForm.Fields["subform1[0]"] as
PdfLoadedXfaForm).Fields["text[0]"] as PdfLoadedXfaTextBoxField;
//Set the width and height of the field
loadedText.Width = 200;
loadedText.Height = 30;
//Set the caption text
loadedText.Caption.Text = "Second Name";
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}

```

Clear XFA date time field value

The Essential PDF supports clearing XFA form date time fields values, you can clear the date time values by using the [ClearValue](#) method available in the [PdfLoadedXfaDateTimeField](#).

The following sample illustrates how to clear the date time field in an existing XFA document.

C#

```

//Load the existing XFA PDF document
PdfLoadedXfaDocument loadedXfaDocument = new
PdfLoadedXfaDocument("input.pdf");
//Get the date time field
PdfLoadedXfaDateTimeField dateTimeField =
loadedXfaDocument.XfaForm.TryGetFieldByCompleteName("form[0].subform[0].date
Time[0]") as PdfLoadedXfaDateTimeField;
//Clear the date time field value
dateTimeField.ClearValue();
//Save and close the document

```

```
loadedXfaDocument.Save("output.pdf");
```

VB.NET

```
'Load the existing XFA PDF document
Dim loadedXfaDocument As PdfLoadedXfaDocument = New
PdfLoadedXfaDocument("input.pdf")
'Get the date time field
Dim dateTimeField As PdfLoadedXfaDateTimeField =
TryCast(loadedXfaDocument.XfaForm.TryGetFieldByCompleteName("form[0].subform
[0].dateTime[0]"), PdfLoadedXfaDateTimeField)
'Clear the date time field value
dateTimeField.ClearValue()
'Save and close the document
loadedXfaDocument.Save("output.pdf")
```

UWP

```
//Load the PDF document as stream
Stream docStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Get the date time field
PdfLoadedXfaDateTimeField dateTimeField =
loadedDocument.XfaForm.TryGetFieldByCompleteName("form[0].subform[0].dateTim
e[0]") as PdfLoadedXfaDateTimeField;
//Clear the date time field value
dateTimeField.ClearValue();
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close();
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Get the date time field
PdfLoadedXfaDateTimeField dateTimeField =
loadedDocument.XfaForm.TryGetFieldByCompleteName("form[0].subform[0].dateTim
e[0]") as PdfLoadedXfaDateTimeField;
//Clear the date time field value
dateTimeField.ClearValue();
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
```

```
//Close the document
loadedDocument.Close();
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
//Load the existing XFA document
PdfLoadedXfaDocument loadedDocument = new PdfLoadedXfaDocument(docStream);
//Get the date time field
PdfLoadedXfaDateTimeField dateTimeField =
loadedDocument.XfaForm.TryGetFieldByCompleteName("form[0].subform[0].dateTim
e[0]") as PdfLoadedXfaDateTimeField;
//Clear the date time field value
dateTimeField.ClearValue();
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close();
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("XfaForm.pdf",
"application/pdf", fdfStream);
}
```

Supported fields for XFA form

The following XFA fields are supported in creating, filling, and flattening the PDF document.

- Text box field
- Numeric field
- DateTime field
- Combo box field
- Radio button field

- List box field
- Check box field
- Text Element
- Button
- Line
- Rectangle
- Circle
- Image

Note: Removing and modifying the form fields is not supported in **static** XFA forms.

Merge Documents

Essential PDF supports merging multiple PDF documents from disk and stream.

Merging multiple documents from disk and stream

You can merge the multiple PDF document using [Merge](#) method of [PdfDocumentBase](#) class, by specifying the path of the documents in a string array.

Refer to the following code example to merge multiple documents from disk.

C#

```
//Creates a new PDF document
PdfDocument finalDoc = new PdfDocument();
// Creates a string array of source files to be merged.
string[] source = { "file1.pdf", "file2.pdf" };
// Merges PDFDocument.
PdfDocument.Merge(finalDoc, source);
//Saves the final document
finalDoc.Save("Sample.pdf");
//Closes the document
finalDoc.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim finalDoc As New PdfDocument()
' Creates a string array of source files to be merged.
Dim source As String() = {"file1.pdf", "file2.pdf"}
' Merges PDFDocument.
PdfDocument.Merge(finalDoc, source)
'Saves the final document
finalDoc.Save("Sample.pdf")
'Closes the document
finalDoc.Close(True)
```

UWP

```
//Due to platform limitations, the PDF file cannot be loaded from disk.
However, you can merge specified document using the following code snippet.
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
```

```
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Merge the document
PdfDocumentBase.Merge(document, loadedDocument);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the documents
document.Close(true);
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respective code samples
Save(stream, "Sample.pdf");
```

ASP.NET CORE

```
//Due to platform limitations, the PDF file cannot be loaded from disk.
However, you can merge multiple documents from stream using the following
code snippet.
//Creates a PDF document
PdfDocument finalDoc = new PdfDocument();
FileStream stream1 = new FileStream("file1.pdf", FileMode.Open,
FileAccess.Read);
FileStream stream2 = new FileStream("file2.pdf", FileMode.Open,
FileAccess.Read);
// Creates a PDF stream for merging
Stream[] streams = { stream1, stream2 };
// Merges PDFDocument.
PdfDocumentBase.Merge(finalDoc, streams);
//Save the document into stream
MemoryStream stream = new MemoryStream();
finalDoc.Save(stream);
stream.Position = 0;
//Close the document
finalDoc.Close(true);
//Disposes the streams.
stream1.Dispose();
stream2.Dispose();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```

//Due to platform limitations, the PDF file cannot be loaded from disk.
However, you can merge multiple documents from stream using the following
code snippet.
//Loads the file as stream
Stream stream1 =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
file1.pdf");
Stream stream2 =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
file2.pdf");
//Creates a PDF stream for merging
Stream[] source = { stream1, stream2 };
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Merge the documents
PdfDocumentBase.Merge(document, source);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the documents
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pdf",
"application/pdf", stream);
}

```

You can merge the PDF document streams by using the following code example.

C#

```

//Creates a PDF document
PdfDocument finalDoc = new PdfDocument();
Stream stream1 = File.OpenRead("file1.pdf");
Stream stream2 = File.OpenRead("file2.pdf");
// Creates a PDF stream for merging.
Stream[] streams = { stream1, stream2 };
// Merges PDFDocument.
PdfDocumentBase.Merge(finalDoc, streams);
//Saves the document
finalDoc.Save("sample.pdf");
//Closes the document
finalDoc.Close(true);
//Disposes the streams
stream1.Dispose();
stream2.Dispose();

```


VB.NET

```

'Creates a PDF document
Dim finalDoc As New PdfDocument()
Dim stream1 As Stream = File.OpenRead("file1.pdf")
Dim stream2 As Stream = File.OpenRead("file2.pdf")
' Creates a PDF stream for merging.
Dim streams As Stream() = {stream1, stream2}
' Merges PDFDocument.
PdfDocumentBase.Merge(finalDoc, streams)
'Saves the document
finalDoc.Save("sample.pdf")
'Closes the document
finalDoc.Close(True)
'Disposes the streams
stream1.Dispose()
stream2.Dispose()

```

UWP

```

//PDF supports merging multiple documents from stream only in Windows Forms,
WPF, ASP.NET, ASP.NET MVC, ASP.NET Core and Xamarin platforms. However, you
can merge specified document using the following code snippet.
//Load the PDF document as stream
Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.file1.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(pdfStream);
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Merge the document
PdfDocumentBase.Merge(document, loadedDocument);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the documents
document.Close(true);
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Sample.pdf");

```

ASP.NET CORE

```

//Creates a PDF document
PdfDocument finalDoc = new PdfDocument();
FileStream stream1 = new FileStream("file1.pdf", FileMode.Open,
FileAccess.Read);
FileStream stream2 = new FileStream("file2.pdf", FileMode.Open,
FileAccess.Read);
// Creates a PDF stream for merging
Stream[] streams = { stream1, stream2 };

```

```
// Merges PDFDocument.
PdfDocumentBase.Merge(finalDoc, streams);
//Save the document into stream
MemoryStream stream = new MemoryStream();
finalDoc.Save(stream);
stream.Position = 0;
//Close the document
finalDoc.Close(true);
//Disposes the streams.
stream1.Dispose();
stream2.Dispose();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Loads the file as stream
Stream stream1 =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
file1.pdf");
Stream stream2 =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
file2.pdf");
//Creates a PDF stream for merging
Stream[] source = { stream1, stream2 };
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Merge the documents
PdfDocumentBase.Merge(document, source);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the documents
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pdf",
"application/pdf", stream);
}
```

Importing pages from multiple documents

Essential PDF provides support for importing the pages from one document to another document using [ImportPage](#) method. The following code illustrates this. The imported page is added to the end of the original document.

C#

```
//Loads document
PdfLoadedDocument lDoc = new PdfLoadedDocument("file1.pdf");
//Creates a new document
PdfDocument document = new PdfDocument();
//Imports the page at 1 from the lDoc
document.ImportPage(lDoc, 1);
//Saves the document
document.Save("sample.pdf");
//Closes the document
document.Close(true);
lDoc.Close(true)
```

VB.NET

```
'Loads document
Dim lDoc As New PdfLoadedDocument("file1.pdf")
'Creates a new document
Dim document As New PdfDocument()
'Imports the page at 1 from the lDoc
document.ImportPage(lDoc, 1)
'Saves the document
document.Save("sample.pdf")
'Closes the document
document.Close(True)
lDoc.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await lDoc.OpenAsync(file);
//Creates a new document
PdfDocument document = new PdfDocument();
//Imports the page at 1 from the lDoc
document.ImportPage(lDoc, 1);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Closes the document
document.Close(true);
lDoc.Close(true);
```

```
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples
Save(stream, "sample.pdf");
```

ASP.NET CORE

```
//Load the PDF document.
FileStream docStream = new FileStream("file1.pdf", FileMode.Open,
FileAccess.Read);
//Load the PDF document.
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Creates a new document
PdfDocument document = new PdfDocument();
//Imports the page at 1 from the lDoc
document.ImportPage(lDoc, 1);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
lDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Creates a new document
PdfDocument document = new PdfDocument();
//Imports the page at 1 from the lDoc
document.ImportPage(lDoc, 1);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
lDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
```

```

}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
    "application/pdf", stream);
}

```

You can import multiple pages from an existing document by using [ImportPageRange](#) method. The following code example illustrates this.

C#

```

//Loads PDF document
PdfLoadedDocument lDoc = new PdfLoadedDocument("file1.pdf");
//Creates a new document
PdfDocument document = new PdfDocument();
//Imports the pages from the lDoc
document.ImportPageRange(lDoc, 0, lDoc.Pages.Count - 1);
//Saves the document
document.Save("sample.pdf");
//Closes the documents
document.Close(true);
lDoc.Close(true);

```

VB.NET

```

'Loads PDF document
Dim lDoc As New PdfLoadedDocument("file1.pdf")
'Creates a new document
Dim document As New PdfDocument()
'Imports the pages from the lDoc
document.ImportPageRange(lDoc, 0, lDoc.Pages.Count - 1)
'Saves the document
document.Save("sample.pdf")
'Closes the documents
document.Close(True)
lDoc.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await lDoc.OpenAsync(file);
//Creates a new document
PdfDocument document = new PdfDocument();
//Imports the page at 1 from the lDoc
document.ImportPageRange(lDoc, 0, lDoc.Pages.Count - 1);
//Save the PDF document to stream

```

```

MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Closes the document
document.Close(true);
lDoc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "sample.pdf");

```

ASP.NET CORE

```

//Load the PDF document.
FileStream docStream = new FileStream("file1.pdf", FileMode.Open,
FileAccess.Read);
//Load the PDF document.
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Creates a new document
PdfDocument document = new PdfDocument();
//Imports the page at 1 from the lDoc
document.ImportPageRange(lDoc, 0, lDoc.Pages.Count - 1);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
lDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
file1.pdf");
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Creates a new document
PdfDocument document = new PdfDocument();
//Imports the page at 1 from the lDoc
document.ImportPageRange(lDoc, 0, lDoc.Pages.Count - 1);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
lDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

You can also import pages from multiple documents and arrange the pages as required. The following code example explains the same.

C#

```

//Loads document
PdfLoadedDocument lDoc = new PdfLoadedDocument("file1.pdf");
//Loads document
PdfLoadedDocument lDoc2 = new PdfLoadedDocument("file1.pdf");
//Creates the new document
PdfDocument document = new PdfDocument();
//Imports and arranges the pages
document.ImportPage(lDoc, 2);
document.ImportPage(lDoc2, 1);
//Saves the document
document.Save("sample.pdf");
//Closes the documents
document.Close(true);
lDoc.Close(true);
lDoc2.Close(true);

```

VB.NET

```

'Loads a document
Dim lDoc As New PdfLoadedDocument("file1.pdf")
Dim lDoc2 As New PdfLoadedDocument("file2.pdf")
'create a new document
Dim document As New PdfDocument()
'imports and arranges the pages
document.ImportPage(lDoc, 2)
document.ImportPage(lDoc2, 1)
'Saves the document
document.Save("sample.pdf")
'Closes the documents
document.Close(True)
lDoc.Close(True)
lDoc2.Close(True)

```

UWP

```

//Load the PDF document as stream

```

```

Stream pdfStream1 =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.file1.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc = new PdfLoadedDocument(pdfStream1);
//Load the PDF document as stream
Stream pdfStream2 =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.file2.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc2 = new PdfLoadedDocument(pdfStream2);
//Creates the new document
PdfDocument document = new PdfDocument();
//Imports and arranges the pages
document.ImportPage(lDoc, 2);
document.ImportPage(lDoc2, 1);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Closes the documents
document.Close(true);
lDoc.Close(true);
lDoc2.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "sample.pdf");

```

ASP.NET CORE

```

//Load the PDF document.
FileStream docStream1 = new FileStream("file1.pdf", FileMode.Open,
FileAccess.Read);
//Load the PDF document.
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream1);
//Load the PDF document.
FileStream docStream2 = new FileStream("file2.pdf", FileMode.Open,
FileAccess.Read);
//Load the PDF document.
PdfLoadedDocument lDoc2 = new PdfLoadedDocument(docStream2);
//Creates the new document
PdfDocument document = new PdfDocument();
//Imports and arranges the pages
document.ImportPage(lDoc, 1);
document.ImportPage(lDoc2, 0);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the documents
document.Close(true);
lDoc.Close(true);
lDoc2.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "sample.pdf";

```



```
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream1 =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
file1.pdf");
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream1);
//Load the file as stream
Stream docStream2 =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
file2.pdf");
PdfLoadedDocument lDoc2 = new PdfLoadedDocument(docStream2);
//Creates the new document
PdfDocument document = new PdfDocument();
//Imports and arranges the pages
document.ImportPage(lDoc, 1);
document.ImportPage(lDoc2, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the documents
document.Close(true);
lDoc.Close(true);
lDoc2.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
sample
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}
```

Best practices

Merging multiple large PDF documents can lead to high runtime memory. So, you can split the documents into multiple documents and later you can merge. This method avoids the extensive memory usage and increases the performance.

Note: Note: The parent PDF document has all the contents in run time memory. It releases the memory once the final PDF document instance is disposed.

You can split a large PDF document into multiple documents using [Split](#) method of [PdfLoadedDocument](#) class. The following code snippet explains this.

C#

```
//Loads the PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("large.pdf");
//Splits the document
loadedDocument.Split("split.pdf");
//Closes the document
loadedDocument.Close(true);
```

VB.NET

```
'Loads the PDF document
Dim loadedDocument As New PdfLoadedDocument("large.pdf")
'Splits the document
loadedDocument.Split("split.pdf")
'Closes the document
loadedDocument.Close(True)
```

UWP

//Due to platform limitations, multiple PDF files cannot be saved to disk. So, Essential PDF supports splitting the document into multiple documents only in Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms.

ASP.NET CORE

//Due to platform limitations, multiple PDF files cannot be saved to disk. So, Essential PDF supports splitting the document into multiple documents only in Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms.

XAMARIN

//Due to platform limitations, multiple PDF files cannot be saved to disk. So, Essential PDF supports splitting the document into multiple documents only in Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms.

The following code shows how to merge multiple PDF documents.

C#

```
//Input documents
string[] inputDocuments = Directory.GetFiles("../Data/Split");
//Creates a PDF document
PdfDocument document = new PdfDocument();
//Merges the document
PdfDocumentBase.Merge(document, inputDocuments);
//Saves and closes the document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Input documents
Dim inputDocuments As String() = Directory.GetFiles("../Data/Split")
```

```

'Creates a PDF document
Dim document As New PdfDocument()
'Merges the document
PdfDocumentBase.Merge(document, inputDocuments)
'Saves and closes the document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Due to platform limitations, the PDF file cannot be loaded from disk.
However, you can merge specified document using the following code snippet.
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Merge the document
PdfDocumentBase.Merge(document, loadedDocument);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the documents
document.Close(true);
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respective code samples
Save(stream, "Sample.pdf");

```

ASP.NET CORE

```

//Due to platform limitations, the PDF file cannot be loaded from disk.
However, you can merge multiple documents from stream using the following code snippet.
//Creates a PDF document
PdfDocument finalDoc = new PdfDocument();
FileStream stream1 = new FileStream("file1.pdf", FileMode.Open, FileAccess.Read);
FileStream stream2 = new FileStream("file2.pdf", FileMode.Open, FileAccess.Read);
// Creates a PDF stream for merging
Stream[] streams = { stream1, stream2 };
// Merges PDFDocument.
PdfDocumentBase.Merge(finalDoc, streams);
//Save the document into stream
MemoryStream stream = new MemoryStream();
finalDoc.Save(stream);
stream.Position = 0;
//Close the document

```

```
finalDoc.Close(true);
//Disposes the streams.
stream1.Dispose();
stream2.Dispose();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Due to platform limitations, the PDF file cannot be loaded from disk.
//However, you can merge multiple documents from stream using the following
//code snippet.
//Loads the file as stream
Stream stream1 =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
file1.pdf");
Stream stream2 =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
file2.pdf");
//Creates a PDF stream for merging
Stream[] source = { stream1, stream2 };
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Merge the documents
PdfDocumentBase.Merge(document, source);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the documents
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.RuntimePlatform == Device.UWP)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pdf",
"application/pdf", stream);
}
```

Optimizing PDF resources when merging PDF documents

Essential PDF provides support to optimize the PDF resources when merging PDF documents. You can optimize the resources by enabling the `OptimizeResources` property available in the `PdfMergeOptions` instance.

Refer to the following code example to optimize the PDF resources when merging PDF documents.

C#

```
//Creates a new PDF document
PdfDocument finalDoc = new PdfDocument();
//Creates a string array of source files to be merged
string[] source = { "file1.pdf", "file2.pdf" };
PdfMergeOptions mergeOptions = new PdfMergeOptions();
//Enable Optimize Resources
mergeOptions.OptimizeResources = true;
//Merges PDFDocument
PdfDocument.Merge(finalDoc, mergeOptions, source);
//Saves the final document
finalDoc.Save("Sample.pdf");
//Closes the document
finalDoc.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim finalDoc As New PdfDocument()
'Creates a string array of source files to be merged
Dim source As String() = {"file1.pdf", "file2.pdf"}
Dim mergeOptions As New PdfMergeOptions()
'Enable Optimize Resources
mergeOptions.OptimizeResources = True
'Merges PDFDocument
PdfDocument.Merge(finalDoc, mergeOptions, source)
'Saves the final document
finalDoc.Save("Sample.pdf")
'Closes the document
finalDoc.Close(True)
```

ASP.NET CORE

```
//Due to platform limitations, the PDF file cannot be loaded from disk.
However, you can optimize PDF resources when merging multiple documents from
stream using the following code snippet
//Creates a PDF document
PdfDocument finalDoc = new PdfDocument();
FileStream stream1 = new FileStream("file1.pdf", FileMode.Open,
FileAccess.Read);
FileStream stream2 = new FileStream("file2.pdf", FileMode.Open,
FileAccess.Read);
//Creates a PDF stream for merging
Stream[] streams = { stream1, stream2 };
PdfMergeOptions mergeOptions = new PdfMergeOptions();
//Enable Optimize Resources
mergeOptions.OptimizeResources = true;
//Merges PDFDocument
PdfDocumentBase.Merge(finalDoc, mergeOptions, streams);
//Save the document into stream
MemoryStream stream = new MemoryStream();
finalDoc.Save(stream);
stream.Position = 0;
```

```
//Close the document
finalDoc.Close(true);
//Disposes the streams
stream1.Dispose();
stream2.Dispose();
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Due to platform limitations, the PDF file cannot be loaded from disk.
//However, you can optimize PDF resources when merging multiple documents from
//stream using the following code snippet
//Loads the file as stream
Stream stream1 =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
file1.pdf");
Stream stream2 =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
file2.pdf");
//Creates a PDF stream for merging
Stream[] source = { stream1, stream2 };
//Create a new PDF document
PdfDocument document = new PdfDocument();
PdfMergeOptions mergeOptions = new PdfMergeOptions();
//Enable Optimize Resources
mergeOptions.OptimizeResources = true;
//Merge the documents
PdfDocumentBase.Merge(document, mergeOptions, source);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the documents
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pdf",
"application/pdf", stream);
}
```

Working with Text Extraction

Essential PDF allows you to extract the text from a particular page or the entire PDF document.

Working with basic text extraction

You can extract the text from a page using [ExtractText](#) method in [PdfPageBase](#) class.

The following code snippet explains how to extract the texts from a page.

C#

```
//Load an existing PDF.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Load the first page.
PdfPageBase page = loadedDocument.Pages[0];
//Extract text from first page.
string extractedText = page.ExtractText();
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load an existing PDF.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Load the first page.
Dim page As PdfPageBase = loadedDocument.Pages(0)
'Extract the text from first page.
Dim extractedText As String = page.ExtractText()
'close the document.
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Load the first page.
PdfPageBase page = loadedDocument.Pages[0];
//Extract text from first page.
string extractedText = page.ExtractText();
//Close the document.
loadedDocument.Close(true);
```

ASP.NET CORE

```
//Load the PDF document.
FileStream docStream = new FileStream("Sample.pdf", FileMode.Open,
FileAccess.Read);
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
```

```
//Load the first page.
PdfPageBase page = loadedDocument.Pages[0];
//Extract text from first page.
string extractedText = page.ExtractText();
//Close the document.
loadedDocument.Close(true);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the first page.
PdfPageBase page = loadedDocument.Pages[0];
//Extract text from first page.
TextLineCollection lineCollection = new TextLineCollection();
string extractedText = page.ExtractText(out lineCollection);
//Close the document
loadedDocument.Close(true);
```

Note: In this method, the text is extracted in the order in which it is written in the document stream and it may not be in the order in which it is viewed in the PDF reader application.

Note: Extracting text from the PDF document pages will not load the entire document content into memory.

The below code illustrates how to extract the text from entire PDF document:

C#

```
// Load an existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
// Loading page collections
PdfLoadedPageCollection loadedPages = loadedDocument.Pages;
string extractedText = string.Empty;
// Extract text from existing PDF document pages
foreach (PdfLoadedPage loadedPage in loadedPages)
{
    extractedText += loadedPage.ExtractText();
}
// Close the document.
loadedDocument.Close(true);
```

VB.NET

```
' Load an existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
' Loading page collections
Dim loadedPages As PdfLoadedPageCollection = loadedDocument.Pages
Dim extractedText As String = String.Empty
' Extract text from existing PDF document pages
For Each loadedPage As PdfLoadedPage In loadedPages
    extractedText &= loadedPage.ExtractText()
```



```

Next loadedPage
' Close the document.
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
// Loading page collections
PdfLoadedPageCollection loadedPages = loadedDocument.Pages;
string extractedText = string.Empty;
// Extract text from existing PDF document pages
foreach (PdfLoadedPage loadedPage in loadedPages)
{
    extractedText += loadedPage.ExtractText();
}
//Close the document.
loadedDocument.Close(true);

```

ASP.NET CORE

```

//Load the PDF document.
FileStream docStream = new FileStream("Sample.pdf", FileMode.Open,
FileAccess.Read);
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
// Loading page collections
PdfLoadedPageCollection loadedPages = loadedDocument.Pages;
string extractedText = string.Empty;
// Extract text from existing PDF document pages
foreach (PdfLoadedPage loadedPage in loadedPages)
{
    extractedText += loadedPage.ExtractText();
}
//Close the document.
loadedDocument.Close(true);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
// Loading page collections
PdfLoadedPageCollection loadedPages = loadedDocument.Pages;
TextLineCollection lineCollection = new TextLineCollection();

```

```
string extractedText = string.Empty;
// Extract text from existing PDF document pages
foreach (PdfLoadedPage loadedPage in loadedPages)
{
    extractedText += loadedPage.ExtractText(out lineCollection);
}
//Close the document
loadedDocument.Close(true);
```

Working with layout based text extraction

You can extract text from the given PDF page based on its layout using [ExtractText\(bool\)](#) overload. In this method, the text is extracted in the layout as it is viewed in the reader application.

Please refer the following code snippet to extract the text with layout.

C#

```
//Load an existing PDF.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Load first page.
PdfPageBase page = loadedDocument.Pages[0];
//Extract text from first page.
string extractedTexts = page.ExtractText(true);
//close the document
loadedDocument.Close(true);
```

VB.NET

```
//Load an existing PDF.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Load first page.
PdfPageBase page = loadedDocument.Pages[0];
//Extract text from first page.
string extractedTexts = page.ExtractText(true);
//close the document
loadedDocument.Close(true);
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Load first page.
PdfPageBase page = loadedDocument.Pages[0];
//Extract text from first page.
string extractedTexts = page.ExtractText(true);
//Close the document.
loadedDocument.Close(true);
```

ASP.NET CORE

```
//Load the PDF document.
FileStream docStream = new FileStream("Sample.pdf", FileMode.Open,
FileAccess.Read);
//Load the PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load first page.
PdfPageBase page = loadedDocument.Pages[0];
//Extract text from first page.
string extractedTexts = page.ExtractText(true);
//Close the document.
loadedDocument.Close(true);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Closes the document
loadedDocument.Close(true);
```

XAMARIN

```
//PDF supports extract text from the given PDF page based on its layout
using ExtractText(bool) overload only in Windows Forms, WPF, ASP.NET,
ASP.NET MVC, ASP.NET Core and UWP platforms.
```

Note: Layout based text extraction may take additional processing time when compared to the normal extraction mode.

Text Extraction with Bounds

Working with Lines

You can get the line and its properties that contains texts by using **TextLine**. Refer to the following code sample.

C#

```
// Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
// Get the first page of the loaded PDF document
PdfPageBase page = loadedDocument.Pages[0];
TextLines lineCollection = new TextLines();
// Extract text from the first page
string extractedText = page.ExtractText(out lineCollection);
// Gets specific line from the collection
TextLine line = lineCollection[0];
// Gets bounds of the line
RectangleF lineBounds = line.Bounds;
// Gets text in the line
string text = line.Text;
```

VB.NET

```
' Load the existing PDF document
```

```

Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument(fileName)
' Get the first page of the loaded PDF document
Dim page As PdfPageBase = loadedDocument.Pages(0)
Dim lineCollection As TextLines = New TextLines()
' Extract text from the first page
Dim extractedText As String = page.ExtractText(lineCollection)
' Gets specific line from the collection
Dim line As TextLine = lineCollection(0)
' Gets bounds of the line
Dim lineBounds As RectangleF = line.Bounds
' Gets text in the line
Dim text As String = line.Text

```

UWP

```

//PDF supports getting the lines and its properties using TextLine only in
WinForms, WPF and Xamarin platforms.

```

ASP.NET CORE

```

//PDF supports getting the lines and its properties using TextLine only in
WinForms, WPF and Xamarin platforms.

```

XAMARIN

```

//Load the existing PDF document
PdfLoadedDocument m_loadedDocument = new PdfLoadedDocument(stream);
//Get the first page of the loaded PDF document
PdfPageBase page = m_loadedDocument.Pages[0];
TextLineCollection lineCollection = new TextLineCollection();
//Extract text from the first page
string m_extractedText = page.ExtractText(out lineCollection);
//Gets specific line from the collection
TextLine line = lineCollection.TextLine[0];
//Gets bounds of the line
RectangleF lineBounds = line.Bounds;
//Gets the font of the text
string fontName = line.FontName;
//Gets the size of the text
float fontSize = line.FontSize;
//Gets font style of the text
FontStyle fontStyle = line.FontStyle;
//Gets text in the line
string text = line.Text;
//Gets collection of the words in the line
List<TextWord> textWordCollection = line.WordCollection;

```

Working with words

You can get the single word and its properties by using **TextWord**. Refer to the following code sample.

C#

```

// Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
// Get the first page of the loaded PDF document

```

```

PdfPageBase page = loadedDocument.Pages[0];
TextLines lineCollection = new TextLines();
// Extract text from the first page
string extractedText = page.ExtractText(out lineCollection);
// Gets specific line from the collection
TextLine line = lineCollection[0];
// Gets bounds of the line
RectangleF lineBounds = line.Bounds;
// Gets text in the line
string text = line.Text;
// Gets collection of the words in the line
List<TextWord> textWordCollection = line.WordCollection;

```

VB.NET

```

' Load the existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument(fileName)
' Get the first page of the loaded PDF document
Dim page As PdfPageBase = loadedDocument.Pages(0)
Dim lineCollection As TextLines = New TextLines()
' Extract text from the first page
Dim extractedText As String = page.ExtractText(lineCollection)
' Gets specific line from the collection
Dim line As TextLine = lineCollection(0)
' Gets bounds of the line
Dim lineBounds As RectangleF = line.Bounds
' Gets text in the line
Dim text As String = line.Text
' Gets collection of the words in the line
Dim textWordCollection As List(Of TextWord) = line.WordCollection

```

UWP

```

//PDF supports getting the word and its properties using TextWord only in
WinForms, WPF and Xamarin platforms.

```

ASP.NET CORE

```

//PDF supports getting the word and its properties using TextWord only in
WinForms, WPF and Xamarin platforms.

```

XAMARIN

```

//Load the existing PDF document
PdfLoadedDocument m_loadedDocument = new PdfLoadedDocument(stream);
//Get the first page of the loaded PDF document
PdfPageBase page = m_loadedDocument.Pages[0];
TextLineCollection lineCollection = new TextLineCollection();
//Extract text from the first page
string m_extractedText = page.ExtractText(out lineCollection);
//Gets specific line from the collection
TextLine line = lineCollection.TextLine[0];
//Gets collection of the words in the line
List<TextWord> textWordCollection = line.WordCollection;
//Gets word from the collection using index

```

```

TextWord textWord = textWordCollection[0];
//Gets bounds of the word
RectangleF WordBounds = textWord.Bounds;
//Gets font name of the word
string wordFontName = textWord.FontName;
//Gets size of the word
float wordFontSize = textWord.FontSize;
//Gets font style of the word
FontStyle wordFontStyle = textWord.FontStyle;
// Gets the word
string wordText = textWord.Text;
// Gets Glyph details of the word
List<TextGlyph> textGlyphCollection = textWord.Glyphs;

```

Working with characters

You can get single character and its properties by using [TextGlyph](#). Refer to the following code sample.

C#

```

//PDF supports getting the single character and its properties using
TextGlyph only in Xamarin platform.

```

VB.NET

```

//PDF supports getting the single character and its properties using
TextGlyph only in Xamarin platform.

```

UWP

```

//PDF supports getting the single character and its properties using
TextGlyph only in Xamarin platform.

```

ASP.NET CORE

```

//PDF supports getting the single character and its properties using
TextGlyph only in Xamarin platform.

```

XAMARIN

```

//Load the existing PDF document
PdfLoadedDocument m_loadedDocument = new PdfLoadedDocument(stream);
//Get the first page of the loaded PDF document
PdfPageBase page = m_loadedDocument.Pages[0];
TextLineCollection lineCollection = new TextLineCollection();
//Extract text from the first page
string m_extractedText = page.ExtractText(out lineCollection);
//Gets specific line from the collection
TextLine line = lineCollection.TextLine[0];
//Gets collection of the words in the line
List<TextWord> textWordCollection = line.WordCollection;
//Gets word from the collection using index
TextWord textWord = textWordCollection[0];
// Gets Glyph details of the word
List<TextGlyph> textGlyphCollection = textWord.Glyphs;

```

```
//Gets character of the word
TextGlyph textGlyph = textGlyphCollection[0];
//Gets bounds of the character
RectangleF glyphBounds = textGlyph.Bounds;
//Gets font name of the character
string GlyphFontName = textGlyph.FontName;
//Gets font size of the character
float GlyphFontSize = textGlyph.FontSize;
//Gets font style of the character
FontStyle GlyphFontStyle = textGlyph.FontStyle;
//Gets character in the word
char GlyphText = textGlyph.Text;
```

Working with Image Extraction

The Essential PDF provides support to extract images from a particular page or an entire PDF document. You can extract the images from a page using the [ExtractImages](#) method in the [PdfPageBase](#) class.

Refer to the following code snippet to extract the images from a PDF page.

C#

```
//Load an existing PDF
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Load the first page
PdfPageBase pageBase = loadedDocument.Pages[0];
//Extract images from first page
Image[] extractedImages = pageBase.ExtractImages();
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load an existing PDF
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Load the first page
Dim pageBase As PdfPageBase = loadedDocument.Pages(0)
'Extract images from first page
Dim extractedImages As Image() = pageBase.ExtractImages()
'Close the document
loadedDocument.Close(True)
```

UWP

```
//PDF supports extracting the images from PDF document only in Windows
Forms, WPF, ASP.NET, and ASP.NET MVC platforms
```

ASP.NET CORE

```
//Load an existing PDF
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the first page
PdfPageBase pageBase = loadedDocument.Pages[0];
//Extract images from first page
```

```
Image[] extractedImages = pageBase.ExtractImages();
//Close the document
loadedDocument.Close(true);
```

XAMARIN

```
//PDF supports extracting the images from PDF document only in Windows
Forms, WPF, ASP.NET, and ASP.NET MVC platforms
```

Note: To extract the images from PDF page in .NET Core, you need to include [Syncfusion.Pdf.Imaging.Portable](#) assembly reference in .NET Core project.

Image informations

To extract the image properties such as bounds, image index, and more from a page, you can use the [ImagesInfo](#) property in the [PdfPageBase](#) class.

Refer to the following code snippet to extract the image info from a PDF page.

C#

```
//Load an existing PDF
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Load the first page
PdfPageBase pageBase = loadedDocument.Pages[0];
//Extracts all the images info from first page
PdfImageInfo[] imagesInfo= pageBase.ImagesInfo;
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load an existing PDF
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Load the first page
Dim pageBase As PdfPageBase = loadedDocument.Pages(0)
'Extracts all the images info from first page
Dim imagesInfo As PdfImageInfo[] = pageBase.ImagesInfo
'Close the document
loadedDocument.Close(True)
```

UWP

```
//PDF supports extracting the images from PDF document only in Windows
Forms, WPF, ASP.NET, and ASP.NET MVC platforms
```

ASP.NET CORE

```
//Load an existing PDF
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the first page
PdfPageBase pageBase = loadedDocument.Pages[0];
//Extracts all the images info from first page
```



```
PdfImageInfo[] imagesInfo= pageBase.GetImagesInfo();  
//Close the document  
loadedDocument.Close(true);
```

XAMARIN

```
//PDF supports extracting the image information from PDF document only in  
Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms
```

Note: To extract the image information from PDF page in .NET Core, you need to include [Syncfusion.Pdf.Imaging.Portable](#) assembly reference in .NET Core project.

Converting HTML to PDF

Essential PDF supports converting HTML pages to PDF document. The converter offers full support for HTML tags, HTML5, CSS3, JavaScript, SVG and page breaks. The following are the three rendering engines:

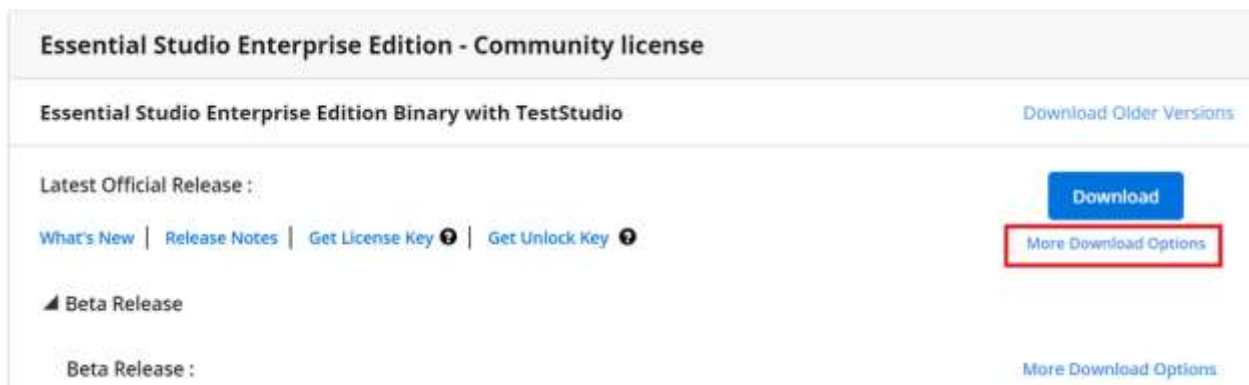
- WebKit rendering
- Blink rendering
- IE rendering

Steps to download the HTML converter installer

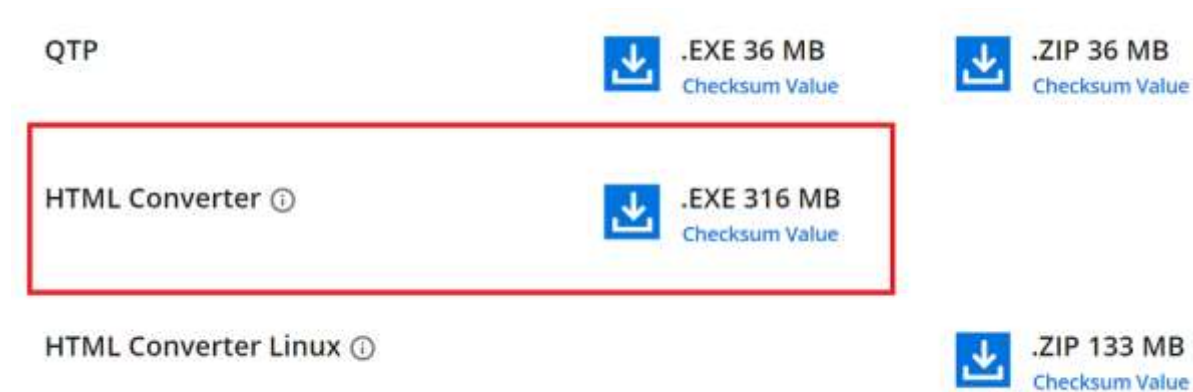
- The latest version of Essential HTML converter can be downloaded from

<https://www.syncfusion.com/downloads/latest-version>

- Click more downloads option from the required product version. Refer to the following screenshot.



- The HTML converter is available under the Add-On section. Refer to the following screenshot.



Getting Started

Essential PDF supports converting HTML contents to PDF. To add the HTML to PDF conversion functionality, add the following assemblies as reference to the project.

Assembly Name	Description
Syncfusion.HtmlConverter.Base	This is required for converting HTML to PDF.
Syncfusion.Pdf.Base	Contains the core feature for creating, manipulating, and saving PDF documents.
Syncfusion.Compression.Base	This is required for compressing the internal contents of a PDF document.

Include the following namespaces in your .cs or .vb file as follows.

C#

```
using Syncfusion.Pdf;
using Syncfusion.HtmlConverter;
```

VB.NET

```
Imports Syncfusion.Pdf
Imports Syncfusion.HtmlConverter
```

ASP.NET CORE

```
using Syncfusion.Pdf;
using Syncfusion.HtmlConverter;
```

Converting HTML to PDF using WebKit rendering engine

To convert website URL or local HTML file to PDF using WebKit rendering engine, refer to the following code snippet. Click the following link for more details to convert the HTML to PDF using WebKit rendering engine.

[Conversion using WebKit Rendering](#)

C#

```
//Initialize the HTML to PDF converter
HtmlToPdfConverter htmlConverter = new
HtmlToPdfConverter(HtmlRenderingEngine.WebKit);
WebKitConverterSettings settings = new WebKitConverterSettings();
//Set WebKit path
settings.WebKitPath = @"QtBinaries/";
//Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = settings;
//Convert URL to PDF
PdfDocument document = htmlConverter.Convert("https://www.google.com");
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Initialize the HTML to PDF converter
Dim htmlConverter As New HtmlToPdfConverter(HtmlRenderingEngine.WebKit)
Dim settings As New WebKitConverterSettings()
'Set WebKit path
settings.WebKitPath = "/QtBinaries/"
'Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = settings
'Convert URL to PDF
Dim document As PdfDocument =
htmlConverter.Convert("https://www.google.com")
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)
```

ASP.NET CORE

```
//Initialize the HTML to PDF converter
HtmlToPdfConverter htmlConverter = new HtmlToPdfConverter();
WebKitConverterSettings settings = new WebKitConverterSettings();
//Set WebKit path
settings.WebKitPath = @"QtBinariesDotNetCore\";
//Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = settings;
//Convert URL to PDF
PdfDocument document = htmlConverter.Convert("https://www.google.com");
FileStream fileStream = new FileStream("Sample.pdf", FileMode.CreateNew,
FileAccess.ReadWrite);
//Save and close the PDF document
document.Save(fileStream);
document.Close(true);
```

Converting HTML to PDF using Blink rendering engine

To convert website URL or local HTML file to PDF using Blink rendering engine, refer to the following code snippet. Click the following link for more details to convert the HTML to PDF using Blink rendering engine.

[Conversion using Blink Rendering](#)

C#

```
//Initialize the HTML to PDF converter with Blink rendering engine
HtmlToPdfConverter htmlConverter = new
HtmlToPdfConverter(HtmlRenderingEngine.Blink);
BlinkConverterSettings blinkConverterSettings = new
BlinkConverterSettings();
//Set the BlinkBinaries folder path
blinkConverterSettings.BlinkPath = @"BlinkBinaries/";
//Assign Blink converter settings to HTML converter
htmlConverter.ConverterSettings = blinkConverterSettings;
//Convert URL to PDF
PdfDocument document = htmlConverter.Convert("https://www.google.com");
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Initialize the HTML to PDF converter with Blink rendering engine
Dim htmlConverter As HtmlToPdfConverter = New
HtmlToPdfConverter(HtmlRenderingEngine.Blink)
Dim blinkConverterSettings As BlinkConverterSettings = New
BlinkConverterSettings()
'Set the BlinkBinaries folder path
blinkConverterSettings.BlinkPath = "BlinkBinaries/"
'Assign Blink converter settings to HTML converter
htmlConverter.ConverterSettings = blinkConverterSettings
'Convert URL to PDF
Dim document As PdfDocument =
htmlConverter.Convert("https://www.google.com")
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)
```

ASP.NET CORE

```
//Currently, Blink rendering engine does not support conversion in .NET Core
platform
```

Converting HTML to PDF using IE rendering engine

To convert website URL or local HTML file to PDF using IE rendering engine, refer to the following code snippet. Click the following link for more details to convert the HTML to PDF using IE rendering engine.

[Conversion using IE Rendering](#)

C#

```
//Initialize the HTML to PDF converter
HtmlToPdfConverter htmlConverter = new
HtmlToPdfConverter(HtmlRenderingEngine.IE);
IEConverterSettings settings = new IEConverterSettings();
//Assign IE settings to HTML converter
htmlConverter.ConverterSettings = settings;
//Convert URL to PDF
PdfDocument document = htmlConverter.Convert("https://www.google.com");
//Save and close the PDF document
```

```
document.Save("Output.pdf");
document.Close(true);
```

VB.NET




























```
'Initialize the HTML to PDF converter
Dim htmlConverter As New HtmlToPdfConverter(HtmlRenderingEngine.IE)
Dim settings As New IEConverterSettings()
'Assign IE settings to HTML converter
htmlConverter.ConverterSettings = settings
'Convert URL to PDF
Dim document As PdfDocument =
htmlConverter.Convert("https://www.google.com")
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)
```






















































ASP.NET CORE

























```
//Currently, IE rendering engine does not support conversion in .NET Core
platform
```

Supported and Unsupported Features by Rendering Engines

The following table shows the WebKit, Blink and IE rendering engines supported features:

Feature	WebKit Renderer	Blink Renderer	IE Renderer
Convert URLs to PDF			
Convert HTML string to PDF			
Images			
Hyperlinks			
CSS			
JavaScript			
ActiveX plugin			
HTML 5			
Page breaks			

Vector Graphics (Selectable/searchable text)			HTML 5 pages are rendered as bitmap.
Handling image and text split across pages			
PDF A1-B			
Tagged PDF			
Page settings			
Header and Footer			
Windows Authentication			
Form Authentication			
HTML to Image			
HTML to SVG			
HTML to MHTML			
SVG to PDF			
HTML Form to PDF Form			
HTTP GET and POST			
Partial HTML to PDF			
Bookmarks			
Repeat HTML Table Header and Footer			
Auto Create Table of Contents			

Windows status			
Print Media Type			
Offline mode conversion			
System proxy			
Manual proxy			
Azure App Service	 (Except free and shared plan)		
Azure Cloud Service			
Azure Function	 (Except consumption plan)		

Working with Document Conversions

Converting Word documents to PDF

Essential PDF allows you to convert a Word document into PDF. For converting a Word document to PDF, the following assemblies need to be referenced in your application

Assembly Name	Description
Syncfusion.DocIO.Base	This assembly has the core features for creating and manipulating Word documents.
Syncfusion.Compression.Base	This assembly is used to package the Word documents
Syncfusion.DocToPdfConverter.Base	This assembly is needed for converting the Word document to PDF.
Syncfusion.Pdf.Base	This assembly has the core features for creating PDF file.
Syncfusion.OfficeChart.Base	This assembly has features to work with chart in Word document.

The following assemblies are need to be referred in addition to the above mentioned assemblies for converting the chart present in the Word document into PDF.

Assembly Name	Description
Syncfusion.OfficeChartToImageConverter.WPF	This assembly is used to convert the chart to image.
Syncfusion.SfChart.WPF	This is supporting assembly for Syncfusion.OfficeChartToImageConverter.WPF
Syncfusion.Shared.WPF	This is supporting assembly for Syncfusion.OfficeChartToImageConverter.WPF

The following namespaces are required to compile the code in this topic.

For Windows Forms, WPF, ASP.NET and ASP.NET MVC applications

- using Syncfusion.OfficeChart
- using Syncfusion.DocIO
- using Syncfusion.DocIO.DLS
- using Syncfusion.DocToPDFConverter
- using Syncfusion.Pdf
- using Syncfusion.OfficeChartToImageConverter

For ASP.NET Core and Xamarin applications

- using Syncfusion.DocIO
- using Syncfusion.DocIO.DLS
- using Syncfusion.DocIORenderer
- using Syncfusion.Pdf

[DocToPDFConverter](#) class is responsible for converting a Word document into PDF. The following code snippet illustrates how to convert a Word document into PDF document.

C#

```
//Load an existing Word document
WordDocument wordDocument = new WordDocument("Template.docx",
FormatType.Docx);
//Initialize chart to image converter for converting charts during Word to
pdf conversion
wordDocument.ChartToImageConverter = new ChartToImageConverter();
//Create an instance of DocToPDFConverter
DocToPDFConverter converter = new DocToPDFConverter();
//Convert Word document into PDF document
PdfDocument pdfDocument = converter.ConvertToPDF(wordDocument);
//Save the PDF file
pdfDocument.Save("WordtoPDF.pdf");
//Close the instance of document objects
pdfDocument.Close(true);
wordDocument.Close();
```


VB.NET

```

'Load an existing Word document
Dim wordDocument As New WordDocument("Template.docx", FormatType.Docx)
'Initialize chart to image converter for converting charts during Word to
pdf conversion
wordDocument.ChartToImageConverter = New ChartToImageConverter()
'Create an instance of DocToPDFConverter
Dim converter As New DocToPDFConverter()
'Convert Word document into PDF document
Dim pdfDocument As PdfDocument = converter.ConvertToPDF(wordDocument)
'Save the PDF file
pdfDocument.Save("WordtoPDF.pdf")
'Close the instance of document objects
pdfDocument.Close(True)
wordDocument.Close()

```

ASP.NET CORE

```

// Open the file as Stream
FileStream docStream = new FileStream(@"Template.docx", FileMode.Open,
FileAccess.Read);
//Loads file stream into Word document
WordDocument wordDocument = new WordDocument(docStream,
Syncfusion.DocIO.FormatTypeAutomatic);
//Instantiation of DocIORenderer for Word to PDF conversion
DocIORenderer render = new DocIORenderer();
//Converts Word document into PDF document
PdfDocument pdfDocument = render.ConvertToPDF(wordDocument);
//Releases all resources used by the Word document and DocIO Renderer
objects
render.Dispose();
wordDocument.Dispose();
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Close the documents
pdfDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = " WordtoPDF.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the Word document as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Template.docx");
//Load the stream into word document
WordDocument wordDocument = new WordDocument(docStream,
Syncfusion.DocIO.FormatTypeAutomatic);

```

```

//Instantiation of DocIORenderer for Word to PDF conversion
DocIORenderer render = new DocIORenderer();
//Converts Word document into PDF document
PdfDocument pdfDocument = render.ConvertToPDF(wordDocument);
//Releases all resources used by the Word document and DocIO Renderer objects
render.Dispose();
wordDocument.Dispose();
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Close the document
pdfDocument.Close();
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer PDF/Xamarin section for respective code samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("WordtoPDF.pdf", "application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("WordtoPDF.pdf", "application/pdf", stream);
}

```

Note:

- Initializing the [ChartToImageConverter](#) is mandatory to convert the charts present in the Word document to PDF. Otherwise the charts will not be exported to the converted PDF.
- [ChartToImageConverter](#) is supported from .NET Framework 4.0 onwards.
- Total number of pages may vary based on unsupported elements in the converted PDF document when compare to Word document.

Customizing the Word document to PDF conversion

Essential DocIO allows you to customize the Word to PDF conversion with the below options:

- Allows to determine the quality of the charts in the converted PDF
- Allows to determine the quality of the JPEG images in the converted PDF
- Allows to reduce the Main Memory usage in Word to PDF conversion by reusing the identical images.

C#

```

//Loads an existing Word document
WordDocument wordDocument = new WordDocument("Sample_Image.docx", FormatType.Docx);
//Initialize chart to image converter for converting charts during Word to pdf conversion

```

```

wordDocument.ChartToImageConverter = new ChartToImageConverter();
//Set the scaling mode for charts
wordDocument.ChartToImageConverter.ScalingMode = ScalingMode.Normal;
//create an instance of DocToPDFConverter - responsible for Word to PDF
conversion
DocToPDFConverter converter = new DocToPDFConverter();
//Set the image quality
converter.Settings.ImageQuality = 100;
//Set the image resolution
converter.Settings.ImageResolution = 640;
//Set true to optimize the memory usage for identical images
converter.Settings.OptimizeIdenticalImages = true;
//Convert Word document into PDF document
PdfDocument pdfDocument = converter.ConvertToPDF(wordDocument);
//Save the PDF file to file system
pdfDocument.Save("WordtoPDF.pdf");
//close the instance of document objects
pdfDocument.Close(true);
wordDocument.Close();

```

VB.NET

```

'Loads an existing Word document
Dim wordDocument As New WordDocument("Sample_Image.docx", FormatType.Docx)
'Initialize chart to image converter for converting charts during Word to
pdf conversion
wordDocument.ChartToImageConverter = New ChartToImageConverter()
'Set the scaling mode for charts
wordDocument.ChartToImageConverter.ScalingMode = ScalingMode.Normal
'create an instance of DocToPDFConverter - responsible for Word to PDF
conversion
Dim converter As New DocToPDFConverter()
'Set the image quality
converter.Settings.ImageQuality = 100
'Set the image resolution
converter.Settings.ImageResolution = 640
'Set true to optimize the memory usage for identical images
converter.Settings.OptimizeIdenticalImages = True
'Convert Word document into PDF document
Dim pdfDocument As PdfDocument = converter.ConvertToPDF(wordDocument)
'Save the PDF file to file system
pdfDocument.Save("WordtoPDF.pdf")
'close the instance of document objects
pdfDocument.Close(True)
wordDocument.Close()

```

ASP.NET CORE

```

//Essential PDF supports customizing the Word document to PDF conversion
only in Windows Forms, WPF, ASP.NET and ASP.NET MVC Platforms.

```

XAMARIN

```

//Essential PDF supports customizing the Word document to PDF conversion
only in Windows Forms, WPF, ASP.NET and ASP.NET MVC Platforms.

```

Converting Excel documents to PDF

[ExcelToPdfConverter](#) is responsible for converting an Excel document into PDF. Essential PDF allows you to convert an entire workbook or a single worksheet into PDF document. For converting an Excel document to PDF, the following assemblies need to be referenced in your application.

- Syncfusion.XlsIO.Base.dll
- Syncfusion.Compression.Base.dll
- Syncfusion.ExcelToPDFConverter.Base.dll
- Syncfusion.ExcelChartToImageConverter.Wpf.dll
- Syncfusion.SfChart.Wpf.dll
- Syncfusion.Shared.Wpf.dll
- Syncfusion.Pdf.Base.dll

Converting a Workbook to PDF

The following code illustrates how to convert a workbook to PDF Document using [ExcelToPdfConverter](#).

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
//Open the Excel Document to Convert
ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
//Initialize PDF Document
PdfDocument pdfDocument = new PdfDocument();
//Convert Excel Document into PDF document
pdfDocument = converter.Convert();
//Save the pdf file
pdfDocument.Save("ExcelToPDF.pdf");
//Dispose the objects
pdfDocument.Close();
converter.Dispose();
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As ExcelEngine = New ExcelEngine
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
'Open the Excel Document to Convert
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Initialize the PDF Document
Dim pdfDocument As PdfDocument = New PdfDocument()
'Convert Excel Document into PDF document
pdfDocument = converter.Convert()
'Save the pdf file
pdfDocument.Save("ExcelToPDF.pdf")
```

```
'Dispose the objects
pdfDocument.Close()
converter.Dispose()
workbook.Close()
excelEngine.Dispose()
```

UWP

```
//Essential PDF supports Excel to PDF conversion only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//Essential PDF supports Excel to PDF conversion only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.
```

XAMARIN

```
//Essential PDF supports Excel to PDF conversion only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.
```

Converting a Worksheet to PDF

The following code shows how to convert a particular sheet to PDF Document using [ExcelToPdfConverter](#).

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
//convert the sheet to pdf
ExcelToPdfConverter converter = new ExcelToPdfConverter(sheet);
PdfDocument pdfDocument= new PdfDocument();
pdfDocument = converter.Convert();
pdfDocument.Save("ExcelToPDF.pdf");
pdfDocument.Close();
converter.Dispose();
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As ExcelEngine = New ExcelEngine
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Converts the particular sheet
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(sheet)
Dim pdfDocument As PdfDocument = New PdfDocument()
```

```
pdfDocument = converter.Convert()
'Save the pdf file
pdfDocument.Save("ExcelToPDF.pdf")
'Dispose the objects
pdfDocument.Close()
converter.Dispose()
workbook.Close()
excelEngine.Dispose()
```

UWP

```
//Essential PDF supports Excel to PDF conversion only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//Essential PDF supports Excel to PDF conversion only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.
```

XAMARIN

```
//Essential PDF supports Excel to PDF conversion only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.
```

Creating individual PDF document for each worksheet

The following code snippet shows how to create an individual PDF document for each worksheet in a workbook.

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
PdfDocument pdfDocument = new PdfDocument();
foreach (IWorksheet sheet in workbook.Worksheets)
{
    ExcelToPdfConverter converter = new ExcelToPdfConverter(sheet);
    pdfDocument = converter.Convert();
    //Save the pdf file
    pdfDocument.Save(sheet.Name+".pdf");
    converter.Dispose();
}
//Dispose the objects
pdfDocument.Close();
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
```

```

Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim pdfDocument As New PdfDocument()
For Each sheet As IWorksheet In workbook.Worksheets
Dim converter As New ExcelToPdfConverter(sheet)
PdfDocument = converter.Convert()
'Save the pdf file
PdfDocument.Save(sheet.Name + ".pdf")
converter.Dispose()
Next
pdfDocument.Close()
workbook.Close()
excelEngine.Dispose()

```

UWP

```

//Essential PDF supports Excel to PDF conversion only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.

```

ASP.NET CORE

```

//Essential PDF supports Excel to PDF conversion only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.

```

XAMARIN

```

//Essential PDF supports Excel to PDF conversion only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.

```

Excel with Chart to PDF

To preserve the charts during Excel to PDF conversion, you should initialize the [ChartToImageConverter](#) of [IApplication](#) interface, otherwise the charts present in worksheet will get skipped. The following code illustrate how to convert an Excel with chart to PDF document.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiating the ChartToImageConverter and
//Assigning the ChartToImageConverter instance of XlsIO application
application.ChartToImageConverter = new ChartToImageConverter();
//Tuning Chart Image Quality
application.ChartToImageConverter.ScalingMode = ScalingMode.Best;
IWorkbook workbook = application.Workbooks.Open("chart.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
PdfDocument pdfDocument = new PdfDocument();
pdfDocument = converter.Convert();
pdfDocument.Save("ExcelToPDF.pdf");
converter.Dispose();
pdfDocument.Close();
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```
Dim excelEngine As New ExcelEngine()  
Dim application As IApplication = excelEngine.Excel  
application.DefaultVersion = ExcelVersion.Excel2013  
'Instantiating the ChartToImageConverter and  
'Assigning the ChartToImageConverter instance of XlsIO application  
application.ChartToImageConverter = New ChartToImageConverter()  
'Tuning Chart Image Quality  
application.ChartToImageConverter.ScalingMode = ScalingMode.Best  
Dim workbook As IWorkbook = application.Workbooks.Open("chart.xlsx")  
Dim worksheet As IWorksheet = workbook.Worksheets(0)  
Dim converter As New ExcelToPdfConverter(workbook)  
Dim pdfDocument As New PdfDocument()  
pdfDocument = converter.Convert()  
pdfDocument.Save("ExcelToPDF.pdf")  
converter.Dispose()  
pdfDocument.Close()  
workbook.Close()  
excelEngine.Dispose()
```

UWP

```
//Essential PDF supports Excel to PDF conversion only in Windows Forms, WPF,  
ASP.NET and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//Essential PDF supports Excel to PDF conversion only in Windows Forms, WPF,  
ASP.NET and ASP.NET MVC platforms.
```

XAMARIN

```
//Essential PDF supports Excel to PDF conversion only in Windows Forms, WPF,  
ASP.NET and ASP.NET MVC platforms.
```

Note: This section is applicable only to the Windows Forms, ASP.NET, MVC and WPF platforms.

Supported Elements

This feature provides support for the following elements:

- Styles
- Character formatting
- Headers and footers
- Images
- Text box
- Hyperlinks
- Document properties
- Comments
- Encryption
- Table Style Support
- Text Rotations

- Excel Page Setup Options
- Unicode Support
- Background Images
- Printing Titles when Converting the Excel to PDF
- Page Break Support
- Print Area Support
- Print Order Support
- Unicode in Headers and Footers

Unsupported Elements

The following list contains unsupported elements that presently will not be preserved in the generated PDF document.

- Grouping columns
- OLE Objects
- Text rotations
- Background images

Converting RTF documents to PDF

Essential PDF allows you to convert a RTF to PDF document. For converting a RTF to PDF, the following assemblies need to be referenced in your application

Assembly Name	Description
Syncfusion.DocIO.Base	This assembly has the core features for creating and manipulating RTF documents.
Syncfusion.Compression.Base	This assembly is used to package the RTF documents
Syncfusion.DocToPdfConverter.Base	This assembly is needed for converting the RTF to PDF.
Syncfusion.Pdf.Base	This assembly has the core features for creating PDF file.

The following namespaces are required to compile the code in this topic.

For Windows Forms, WPF, ASP.NET and ASP.NET MVC applications

- using Syncfusion.DocIO
- using Syncfusion.DocIO.DLS
- using Syncfusion.DocToPDFConverter
- using Syncfusion.Pdf

For ASP.NET Core and Xamarin applications

- using Syncfusion.DocIO

- using Syncfusion.DocIO.DLS
- using Syncfusion.DocIO.Renderer
- using Syncfusion.Pdf

[DocToPDFConverter](#) class is responsible for converting a RTF to PDF. The following code snippet illustrates how to convert a RTF to PDF document.

C#

```
//Load an existing RTF document
WordDocument rtfDocument = new WordDocument(inputFileName);
//Create an instance of DocToPDFConverter
DocToPDFConverter converter = new DocToPDFConverter();
//Convert Word document into PDF document
PdfDocument pdfDocument = converter.ConvertToPDF(rtfDocument);
//Save the PDF file
pdfDocument.Save("RTFToPDF.pdf");
//Close the instance of document objects
pdfDocument.Close(true);
rtfDocument.Close();
```

VB.NET

```
'Load an existing Word document
Dim rtfDocument As New WordDocument(inputFileName)
'Create an instance of DocToPDFConverter
Dim converter As New DocToPDFConverter()
'Convert Word document into PDF document
Dim pdfDocument As PdfDocument = converter.ConvertToPDF(rtfDocument)
'Save the PDF file
pdfDocument.Save("RTFToPDF.pdf")
'Close the instance of document objects
pdfDocument.Close(True)
rtfDocument.Close()
```

ASP.NET CORE

```
//Open the file as Stream
FileStream docStream = new FileStream(@"Input.rtf", FileMode.Open,
FileAccess.Read);
//Loads file stream into Word document
WordDocument wordDocument = new WordDocument(docStream,
Syncfusion.DocIO.FormatType.Automatic);
//Instantiation of DocIO.Renderer for Word to PDF conversion
DocIO.Renderer render = new DocIO.Renderer();
//Converts Word document into PDF document
PdfDocument pdfDocument = render.ConvertToPDF(wordDocument);
//Releases all resources used by the Word document and DocIO Renderer
objects
render.Dispose();
wordDocument.Dispose();
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
```

```
//Close the documents
pdfDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = " RTFTToPDF.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the Word document as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.rtf");
//Load an existing Word document
WordDocument rtfDocument = new WordDocument(docStream,
Syncfusion.DocIO.FormatType.Automatic);
//Instantiation of DocIORenderer for Word to PDF conversion
DocIORenderer render = new DocIORenderer();
//Converts Word document into PDF document
PdfDocument pdfDocument = render.ConvertToPDF(rtfDocument);
//Releases all resources used by the Word document and DocIO Renderer
objects
render.Dispose();
rtfDocument.Dispose();
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Close the document
pdfDocument.Close();
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("RTFTToPDF.pdf"
, "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("RTFTToPDF.pdf",
"application/pdf", stream);
}
```

Note: Total number of pages may vary based on unsupported elements in the converted PDF document when compare to RTF document.

[Customizing the RTF to PDF conversion](#)

Essential DocIO allows you to customize the RTF to PDF conversion with the below options:

- Allows to determine the quality of the JPEG images in the converted PDF
- Allows to reduce the Main Memory usage in RTF to PDF conversion by reusing the identical images.

C#

```
//Loads an existing Word document
WordDocument rtfDocument = new WordDocument(inputFileName);
//create an instance of DocToPDFConverter - responsible for Word to PDF
conversion
DocToPDFConverter converter = new DocToPDFConverter();
//Set the image quality
converter.Settings.ImageQuality = 100;
//Set the image resolution
converter.Settings.ImageResolution = 640;
//Set true to optimize the memory usage for identical images
converter.Settings.OptimizeIdenticalImages = true;
//Convert Word document into PDF document
PdfDocument pdfDocument = converter.ConvertToPDF(rtfDocument);
//Save the PDF file to file system
pdfDocument.Save("RTFToPDF.pdf");
//close the instance of document objects
pdfDocument.Close(true);
rtfDocument.Close();
```

VB.NET

```
'Loads an existing Word document
Dim rtfDocument As New WordDocument(inputFileName)
'create an instance of DocToPDFConverter - responsible for Word to PDF
conversion
Dim converter As New DocToPDFConverter()
'Set the image quality
converter.Settings.ImageQuality = 100
'Set the image resolution
converter.Settings.ImageResolution = 640
'Set true to optimize the memory usage for identical images
converter.Settings.OptimizeIdenticalImages = True
'Convert Word document into PDF document
Dim pdfDocument As PdfDocument = converter.ConvertToPDF(rtfDocument)
'Save the PDF file to file system
pdfDocument.Save("RTFToPDF.pdf")
'close the instance of document objects
pdfDocument.Close(True)
rtfDocument.Close()
```

ASP.NET CORE

```
//Essential PDF supports customizing the RTF to PDF conversion only Windows
Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

XAMARIN

```
//Essential PDF supports customizing the RTF to PDF conversion only Windows
Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

Converting TIFF to PDF

Converting multi page TIFF to PDF

Multi frame TIFF image can be converted to PDF document. This can be done by accessing each frame of the multi frame TIFF image and rendering it in each page of the PDF document.

The code snippet to illustrate the same is given below.

C#

```
//Create a PDF document
PdfDocument pdfDocument = new PdfDocument();
//Load multi frame TIFF image
PdfBitmap tiffImage = new PdfBitmap("image.tiff");
//Get the frame count
int frameCount = tiffImage.FrameCount;
//Access each frame and draw into the page
for (int i = 0; i < frameCount; i++)
{
    tiffImage.ActiveFrame = i;
    //Add a section to the PDF document
    PdfSection section = pdfDocument.Sections.Add();
    //Set page margins
    section.PageSettings.Margins.All = 0;
    //Create a PDF unit converter instance
    PdfUnitConvertor converter = new PdfUnitConvertor();
    //Convert to point
   .SizeF size = converter.ConvertFromPixels(tiffImage.PhysicalDimension,
    PdfGraphicsUnit.Point);
    //Set page orientation
    section.PageSettings.Orientation = (size.Width > size.Height) ?
    PdfPageOrientation.Landscape : PdfPageOrientation.Portrait;
    //Set page size
    section.PageSettings.Size = size;
    //Add a page to the section
    PdfPage page = section.Pages.Add();
    //Draw TIFF image into the PDF page
    page.Graphics.DrawImage(tiffImage, PointF.Empty, size);
}
//Save and close the document
pdfDocument.Save("Sample.pdf");
pdfDocument.Close(true);
```

VB.NET

```
'Create a PDF document
Dim pdfDocument As New PdfDocument()
'Load multi frame TIFF image
Dim tiffImage As New PdfBitmap("image.tiff")
'Get the frame count
Dim frameCount As Integer = tiffImage.FrameCount
'Access each frame and draw into the page
For i As Integer = 0 To frameCount - 1
    tiffImage.ActiveFrame = i
'Add a section to the PDF document
```

```

Dim section As PdfSection = pdfDocument.Sections.Add()
'Set page margins
section.PageSettings.Margins.All = 0
'Create a PDF unit converter instance
Dim converter As New PdfUnitConvertor()
'Convert to point
Dim size As.SizeF = converter.ConvertFromPixels(tiffImage.PhysicalDimension,
PdfGraphicsUnit.Point)
'Set page orientation
If size.Width > size.Height Then section.PageSettings.Orientation =
PdfPageOrientation.Landscape
'Set page size
section.PageSettings.Size = size
'Add a page to the section
Dim page As PdfPage = section.Pages.Add()
'Draw TIFF image into the PDF page
page.Graphics.DrawImage(tiffImage, PointF.Empty, size)
Next
'Save and close the document
pdfDocument.Save("Sample.pdf")
pdfDocument.Close(True)

```

UWP

```

//Create a PDF document
PdfDocument pdfDocument = new PdfDocument();
//Load multi frame TIFF image
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.image.tiff");
PdfBitmap tiffImage = new PdfBitmap(imageStream);
//Get the frame count
int frameCount = tiffImage.FrameCount;
//Access each frame and draw into the page
for (int i = 0; i < frameCount; i++)
{
tiffImage.ActiveFrame = i;
//Add a section to the PDF document
PdfSection section = pdfDocument.Sections.Add();
//Set page margins
section.PageSettings.Margins.All = 0;
//Create a PDF unit converter instance
PdfUnitConvertor converter = new PdfUnitConvertor();
//Convert to point
SizeF size = converter.ConvertFromPixels(tiffImage.PhysicalDimension,
PdfGraphicsUnit.Point);
//Set page orientation
section.PageSettings.Orientation = (size.Width > size.Height) ?
PdfPageOrientation.Landscape : PdfPageOrientation.Portrait;
//Set page size
section.PageSettings.Size = size;
//Add a page to the section
PdfPage page = section.Pages.Add();
//Draw TIFF image into the PDF page
page.Graphics.DrawImage(tiffImage, PointF.Empty, size);
}

```

```

MemoryStream memoryStream = new MemoryStream();
//Save the document
await pdfDocument.SaveAsync(memoryStream);
//Close the documents
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(memoryStream, "Sample.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Load the multi frame TIFF image from the disk
FileStream imageStream = new FileStream("image.tiff", FileMode.Open,
FileAccess.Read);
PdfTiffImage tiffImage = new PdfTiffImage(imageStream);
//Get the frame count
int frameCount = tiffImage.FrameCount;
//Access each frame and draw into the page
for (int i = 0; i < frameCount; i++)
{
//Add a section to the PDF document
PdfSection section = document.Sections.Add();
//Set page margins
section.PageSettings.Margins.All = 0;
tiffImage.ActiveFrame = i;
//Create a PDF unit converter instance
PdfUnitConvertor converter = new PdfUnitConvertor();
//Convert to point
Syncfusion.Drawing.SizeF size =
converter.ConvertFromPixels(tiffImage.PhysicalDimension,
PdfGraphicsUnit.Point);
//Set page orientation
section.PageSettings.Orientation = (size.Width > size.Height) ?
PdfPageOrientation.Landscape : PdfPageOrientation.Portrait;
//Set page size
section.PageSettings.Size = size;
//Add a page to the section
PdfPage page = section.Pages.Add();
//Draw TIFF image into the PDF page
page.Graphics.DrawImage(tiffImage, Syncfusion.Drawing.PointF.Empty, size);
}
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";

```

```
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Essential PDF supports converting multi page TIFF to PDF only in Windows
Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms.
```

Note: 1. Essential PDF supports converting TIFF to PDF with [Syncfusion.Pdf.Imaging.Portable](#) assembly reference in ASP.NET Core.

Compression in monochrome images

Essential PDF supports JBIG2 compression for best compression of monochrome images.

Refer the below code snippet to draw a single frame monochrome TIFF image with JBIG2 compression

C#

```
//Create a PDF document
PdfDocument pdfDocument = new PdfDocument();
//Add a page
PdfPage page = pdfDocument.Pages.Add();
//Load single frame TIFF image
PdfBitmap tiffImage = PdfImage.FromFile("image.tiff") as PdfBitmap;
//Set encode type
tiffImage.Encoding = EncodingType.JBIG2;
//Draw an image
page.Graphics.DrawImage(tiffImage, 0, 0, page.GetClientSize().Width,
page.GetClientSize().Height);
//Save and close the document
pdfDocument.Save("Sample.pdf");
pdfDocument.Close(true);
```

VB.NET

```
'Create a PDF document
Dim pdfDocument As New PdfDocument()
'Add a page
Dim page As PdfPage = pdfDocument.Pages.Add()
'Load single frame TIFF image
Dim tiffImage As PdfBitmap = TryCast(PdfImage.FromFile("image.tiff"),
PdfBitmap)
'Set encode type
tiffImage.Encoding = EncodingType.JBIG2
'Draw an image
page.Graphics.DrawImage(tiffImage, 0, 0, page.GetClientSize().Width,
page.GetClientSize().Height)
'Save and close the document
pdfDocument.Save("Sample.pdf")
pdfDocument.Close(True)
```

UWP


```
//Essential PDF supports compressing monochrome images only in Windows Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms.
```

ASP.NET CORE

```
//Essential PDF supports compressing monochrome images only in Windows Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms.
```

XAMARIN

```
//Essential PDF supports compressing monochrome images only in Windows Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms.
```

Note: 1. Currently the JBIG2Decode compression is supported only in lossy mode and also only single frame TIFF images are supported.

2. By default, all monochrome images will be compressed in CITT4 compression.

Converting XPS document to PDF

The XPS (XML Paper Specification) document format is a fixed document format which consists of structured XML markup that defines the layout of a document and the visual appearance of each page, along with rendering rules for distributing, archiving, rendering, processing and printing the documents.

Essential PDF provides support for converting XPS to PDF using [XPSToPdfConverter](#) class.

The below code illustrates how to convert XPS to PDF.

C#

```
//Create converter class
XPSToPdfConverter converter = new XPSToPdfConverter();
//Convert the XPS to PDF
PdfDocument document = converter.Convert(xpsFileName);
//Save and close the document
document.Save("Sample.pdf");
document.Close(true);
```

VB.NET

```
'Create converter class
Dim converter As New XPSToPdfConverter()
'Convert the XPS to PDF
Dim document As PdfDocument = converter.Convert(xpsFileName)
'Save and close the document
document.Save("Sample.pdf")
document.Close(True)
```

UWP

```
//Create converter class
XPSToPdfConverter converter = new XPSToPdfConverter();
//Load the XPS file
Stream fileStream =
    typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.input.xps");
```

```
//Convert the XPS to PDF
PdfDocument document = converter.Convert(fileStream);
MemoryStream memoryStream = new MemoryStream();
//Save the document
await document.SaveAsync(memoryStream);
//Close the documents
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(memoryStream, "Sample.pdf");
```

ASP.NET CORE

```
//Initialize XPS to PDF converter.
XPSToPdfConverter converter = new XPSToPdfConverter();
//Open the XPS file as stream.
FileStream fileStream = new FileStream("Input.xps", FileMode.Open,
FileAccess.ReadWrite);
//Convert the XPS to PDF
PdfDocument document = converter.Convert(fileStream);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into stream.
document.Save(stream);
//If the position is not set to '0' then the PDF will be empty.
stream.Position = 0;
//Close the documents.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

//Essential PDF supports converting XPS document to PDF only in Windows Forms, WPF, ASP.NET, ASP.NET MVC and UWP platforms.

Note: Essential PDF supports converting XPS to PDF with [Syncfusion.XpsToPdfConverter.Net.Core](#) package reference in .NET Core application.

Supported Elements

The below table shows the list of elements supported in XPS during the conversion.

Element	Convert to PDF
ArcSegment	Yes
Canvas	Yes

Element	Convert to PDF
DocumentOutline	No
DocumentReference	No
FigureStructure	No
FixedPageResources	Yes
Glyphs	Yes
Gradient	Yes
ImageBrush (JPG, BMP, GIF, TIFF)	Yes
Intent	Yes
LinkTarget	Yes
ListItemStructure	Yes
ListStructure	Yes
MatrixTransform	Yes
NamedElement	No
OutlineEntry	No
PageContent	Yes
PageContentLinkTargets	No
ParagraphStructure	No
Path	Yes
PolyBezierSegment	Yes
PolyLineSegment	Yes
PolyQuadraticBezierSegment	Yes
ResourceDictionary	Yes

Element	Convert to PDF
SectionStructure	No
SignBy	No
SignatureDefinition	No
SignatureDefinitions	No
SigningLocation	No
SolidColorBrush	Yes
SpotLocation	No
Story	No
TableStructure	No
VisualBrush	No

Converting PDF to Image

PDF pages can be converted into images. To add PDF to image functionality in an application, add the following assemblies as reference to the project:

1. Syncfusion.Compression.Base.dll
2. Syncfusion.Pdf.Base.dll

The following code snippet illustrates how to convert PDF page into image.

C#

```
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("HTTP
Succinctly.pdf");
//Get the PDF page count
int count = loadedDocument.Pages.Count;
//Convert each page to image file
for (int i = 0; i < count; i++)
{
    //Export PDF page to image
    Image image = loadedDocument.ExportAsImage(i);
    //Save the exported image
    image.Save("Image" + i + ".png", System.Drawing.Imaging.ImageFormat.Png);
}
//Close the document
ldoc.Close(true);
```

VB.NET

```

'Load the existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("HTTP
Succinctly.pdf")
'Get the PDF page count
Dim count As Integer = loadedDocument.Pages.Count
'Convert each page to image file
Dim i As Integer = 0
Do While (i < count)
'Export PDF page to image
Dim image As Image = loadedDocument.ExportAsImage(i)
'Save the exported image
image.Save(("Image" _+ (i + ".png")),
System.Drawing.Imaging.ImageFormat.Png)
i = (i + 1)
Loop
'Close the document
ldoc.Close(true)

```

MHTML to PDF

The MHTML file can be converted to PDF using [WebKit rendering engine](https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#prerequisites-for-windows). Please refer the below code snippet,

Prerequisites - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#prerequisites-for-windows>

Troubleshooting - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#troubleshooting>

C#

```

//Initialize HTML converter with WebKit rendering engine
HtmlToPdfConverter htmlConverter = new
HtmlToPdfConverter(HtmlRenderingEngine.WebKit);
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"QtBinaries/";
//Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = webKitSettings;
//Convert MHTML to PDF
PdfDocument document = htmlConverter.Convert("input.mhtml");
//Save the document
document.Save("Sample.pdf");
document.Close();

```

VB.NET

```

'Initialize HTML converter with WebKit rendering engine
Dim htmlConverter As New HtmlToPdfConverter(HtmlRenderingEngine.WebKit)
Dim webKitSettings As New WebKitConverterSettings()
'Set WebKit path
webKitSettings.WebKitPath = "QtBinaries/"
'Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = webKitSettings
'Convert MHTML to PDF

```

```
Dim document As PdfDocument = htmlConverter.Convert("input.mhtml")
'Save the document
document.Save("Sample.pdf")
document.Close()
```

UWP

```
//Essential PDF supports converting MHTML to PDF only in Windows Forms, WPF,
ASP.NET, ASP.NET MVC and ASP.NET Core platforms.
```

ASP.NET CORE

```
//Initialize HTML to PDF converter
HtmlToPdfConverter htmlConverter = new HtmlToPdfConverter();
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"QtBinariesDotNetCore/";
//Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = webKitSettings;
//Convert MHTML to PDF
PdfDocument document = htmlConverter.Convert(@"input.mhtml");
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = " Sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Essential PDF supports converting MHTML to PDF only in Windows Forms, WPF,
ASP.NET, ASP.NET MVC and ASP.NET Core platforms.
```

HTML to MHTML

The [WebKit HTML Converter](#) provides support for converting the webpage to MHTML. Please refer the below code snippet,

Prerequisites - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#prerequisites-for-windows>

Troubleshooting - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#troubleshooting>

C#

```
//Initialize HTML converter with WebKit rendering engine
HtmlToPdfConverter htmlConverter = new
HtmlToPdfConverter(HtmlRenderingEngine.WebKit);
```

```

WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"QtBinaries/";
//Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = webKitSettings;
//Convert URL to MHTML
htmlConverter.ConvertToMhtml("http://www.syncfusion.com", "sample.mhtml");

```

VB.NET

```

'Initialize HTML converter with WebKit rendering engine
Dim htmlConverter As New HtmlToPdfConverter(HtmlRenderingEngine.WebKit)
Dim webKitSettings As New WebKitConverterSettings()
'Set WebKit path
webKitSettings.WebKitPath = "/QtBinaries/"
'Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = webKitSettings
'Convert URL to MHTML
htmlConverter.ConvertToMhtml("http://www.syncfusion.com", "sample.mhtml")

```

UWP

```

//Essential PDF supports converting HTML to MHTML only in Windows Forms,
WPF, ASP.NET, ASP.NET MVC platforms.

```

ASP.NET CORE

```

//Essential PDF supports converting HTML to MHTML only in Windows Forms,
WPF, ASP.NET, ASP.NET MVC platforms.

```

XAMARIN

```

//Essential PDF supports converting HTML to MHTML only in Windows Forms,
WPF, ASP.NET, ASP.NET MVC platforms.

```

HTML to Raster Image

The [WebKit HTML Converter](#) provides support for converting webpage to Image. Please refer the below code snippet,

Prerequisites - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#prerequisites-for-windows>

Troubleshooting - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#troubleshooting>

C#

```

//Initialize HTML converter with WebKit rendering engine
HtmlToPdfConverter htmlConverter = new
HtmlToPdfConverter(HtmlRenderingEngine.WebKit);
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"QtBinaries/";
//Assign WebKit settings to HTML converter

```

```
htmlConverter.ConverterSettings = webKitSettings;
//Convert URL to Image
Image[] image = htmlConverter.ConvertToImage("http://www.syncfusion.com");
//Save the image
image[0].Save("Sample.jpg");
```

VB.NET

```
'Initialize HTML converter with WebKit rendering engine
Dim htmlConverter As New HtmlToPdfConverter(HtmlRenderingEngine.WebKit)
Dim webKitSettings As New WebKitConverterSettings()
'Set WebKit path
webKitSettings.WebKitPath = "/QtBinaries/"
'Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = webKitSettings
'Convert URL to Image
Dim image As Image() =
htmlConverter.ConvertToImage("http://www.syncfusion.com ")
'Save the image
image(0).Save("Sample.jpg")
```

UWP

```
//Essential PDF supports converting HTML to raster image only in Windows
Forms, WPF, ASP.NET, ASP.NET MVC platforms.
```

ASP.NET CORE

```
//Initialize HTML converter with WebKit rendering engine
HtmlToPdfConverter htmlConverter = new HtmlToPdfConverter();
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"QtBinariesDotNetCore/";
//Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = webKitSettings;
//Convert URL to Image
Image image = htmlConverter.ConvertToImage("http://www.google.com");
byte[] imageByte = image.ImageData;
//Save the image
File.WriteAllBytes("Output.jpg", imageByte);
```

XAMARIN

```
//Essential PDF supports converting HTML to raster image only in Windows
Forms, WPF, ASP.NET, ASP.NET MVC platforms.
```

HTML string to Raster Image

The [WebKit HTML Converter](#) provides support for converting HTML string to Image. Please refer the below code snippet,

Prerequisites - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#prerequisites-for-windows>

Troubleshooting - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#troubleshooting>

C#

```
string htmlString = "<html><body>Hello World!!!</body></html>";
string baseUrl = "";
//Initialize HTML converter with WebKit rendering engine
HtmlToPdfConverter htmlConverter = new
HtmlToPdfConverter(HtmlRenderingEngine.WebKit);
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"QtBinaries/";
//Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = webKitSettings;
//Convert HTML string to Image
Image[] image = htmlConverter.ConvertToImage(htmlString, baseUrl);
//Save the image
image[0].Save("Sample.jpg");
```

VB.NET

```
Dim htmlString As String = "<html><body>Hello World!!!</body></html>"
Dim baseUrl As String = ""
'Initialize HTML converter with WebKit rendering engine
Dim htmlConverter As New HtmlToPdfConverter(HtmlRenderingEngine.WebKit)
Dim webKitSettings As New WebKitConverterSettings()
'Set WebKit path
webKitSettings.WebKitPath = "QtBinaries/"
'Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = webKitSettings
'Convert HTML string to Image
Dim image As Image() = htmlConverter.ConvertToImage(htmlString, baseUrl)
'Save the image
image(0).Save("Sample.jpg")
```

UWP

```
//Essential PDF supports converting HTML string to raster image only in
Windows Forms, WPF, ASP.NET, ASP.NET MVC platforms.
```

ASP.NET CORE

```
//Initialize HTML converter with WebKit rendering engine
HtmlToPdfConverter htmlConverter = new HtmlToPdfConverter();
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"QtBinariesDotNetCore/";
//Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = webKitSettings;
//HTML string and Base URL
string htmlString = "<html><body>Hello World!!!</body></html>";
string baseUrl = "";
//Convert HTML string to Image
Image image = htmlConverter.ConvertToImage(htmlString, baseUrl);
```

```
byte[] imageByte = image.ImageData;
//Save the image
File.WriteAllBytes("Output.jpg", imageByte);
```

XAMARIN

```
//Essential PDF supports converting HTML string to raster image only in
Windows Forms, WPF, ASP.NET, ASP.NET MVC platforms.
```

Partial webpage to Raster Image

The [WebKit HTML Converter](#) provides support for converting partial webpage to Image. Please refer the below code snippet,

Prerequisites - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#prerequisites-for-windows>

Troubleshooting - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#troubleshooting>

C#

```
//Initialize HTML converter
HtmlToPdfConverter htmlConverter = new
HtmlToPdfConverter(HtmlRenderingEngine.WebKit);
// WebKit converter settings
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Assign the WebKit binaries path
webKitSettings.WebKitPath = @"QtBinaries/";
//Assign the WebKit settings
htmlConverter.ConverterSettings = webKitSettings;
//Convert Partial HTML to Image
Image[] image = htmlConverter.ConvertPartialHtmlToImage("input.html",
"pic");
//Save Image
image[0].Save("Output.jpg");
```

VB.NET

```
'Initialize HTML converter
Dim htmlConverter As New HtmlToPdfConverter(HtmlRenderingEngine.WebKit)
' WebKit converter settings
Dim webKitSettings As New WebKitConverterSettings()
'Assign the WebKit binaries path
webKitSettings.WebKitPath = "QtBinaries/"
'Assign the WebKit settings
htmlConverter.ConverterSettings = webKitSettings
'Convert Partial HTML to Image
Dim image As Image() = htmlConverter.ConvertPartialHtmlToImage("input.html",
"pic")
'Save Image
image(0).Save("Output.jpg")
```

HTML

```
<html>
```

```

<head>
</head>
<body>
Hello world
<div id="pic">
<br>
This is a Syncfusion Logo
</div>
<div>
Hello world
</div>
</body>
</html>

```

UWP

//Essential PDF supports converting partial webpage to raster image only in Windows Forms, WPF, ASP.NET, ASP.NET MVC platforms.

ASP.NET CORE

```

//Initialize HTML converter with WebKit rendering engine
HtmlToPdfConverter htmlConverter = new HtmlToPdfConverter();
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"QtBinariesDotNetCore/";
//Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = webKitSettings;
//Convert Partial HTML to Image
Image image =
htmlConverter.ConvertPartialHtmlToImage("http://www.google.com", "lga");
byte[] imageByte = image.ImageData;
//Save the image
File.WriteAllBytes("Output.jpg", imageByte);

```

XAMARIN

//Essential PDF supports converting partial webpage to raster image only in Windows Forms, WPF, ASP.NET, ASP.NET MVC platforms.

HTML to SVG

The [WebKit HTML Converter](#) provides support for converting HTML to SVG. Please refer the below code snippet,

Prerequisites - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#prerequisites-for-windows>

Troubleshooting - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#troubleshooting>

C#

```

//Initialize HTML converter with WebKit rendering engine

```

```

HtmlToPdfConverter htmlConverter = new
HtmlToPdfConverter(HtmlRenderingEngine.WebKit);
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();
//Set WebKit path
webKitSettings.WebKitPath = @"QtBinaries/";
//Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = webKitSettings;
//Convert URL to SVG
htmlConverter.ConvertToSvg("http://www.syncfusion.com", "sample.svg");

```

VB.NET

```

'Initialize HTML converter with WebKit rendering engine
Dim htmlConverter As New HtmlToPdfConverter(HtmlRenderingEngine.WebKit)
Dim webKitSettings As New WebKitConverterSettings()
'Set WebKit path
webKitSettings.WebKitPath = "/QtBinaries/"
'Assign WebKit settings to HTML converter
htmlConverter.ConverterSettings = webKitSettings
'Convert URL to SVG
htmlConverter.ConvertToSvg("http://www.syncfusion.com", "sample.svg")

```

UWP

```

//Essential PDF supports converting HTML to SVG only in Windows Forms, WPF,
ASP.NET, ASP.NET MVC platforms.

```

ASP.NET CORE

```

//Essential PDF supports converting HTML to SVG only in Windows Forms, WPF,
ASP.NET, ASP.NET MVC platforms.

```

XAMARIN

```

//Essential PDF supports converting HTML to SVG only in Windows Forms, WPF,
ASP.NET, ASP.NET MVC platforms.

```

Partial webpage to SVG

The [WebKit HTML Converter](#) provides support for converting partial webpage to SVG. Please refer the below code snippet,

Prerequisites - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#prerequisites-for-windows>

Troubleshooting - <https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit#troubleshooting>

C#

```

//Initialize HTML converter
HtmlToPdfConverter htmlConverter = new
HtmlToPdfConverter(HtmlRenderingEngine.WebKit);
// WebKit converter settings
WebKitConverterSettings webKitSettings = new WebKitConverterSettings();

```

```
//Assign the WebKit binaries path
webkitSettings.WebKitPath = @"/QtBinaries/";
//Assign the WebKit settings
htmlConverter.ConverterSettings = webKitSettings;
//Convert Partial HTML to SVG
htmlConverter.ConvertPartialHtmlToSvg("input.html", "pic", "Output.svg");
```

VB.NET

```
'Initialize HTML converter
Dim htmlConverter As New HtmlToPdfConverter(HtmlRenderingEngine.WebKit)
' WebKit converter settings
Dim webKitSettings As New WebKitConverterSettings()
'Assign the WebKit binaries path
webkitSettings.WebKitPath = "/QtBinaries/"
'Assign the WebKit settings
htmlConverter.ConverterSettings = webKitSettings
'Convert Partial HTML to SVG
htmlConverter.ConvertPartialHtmlToSvg("input.html", "pic", "Output.svg")
```

HTML

```
<html>
<head>
</head>
<body>
Hello world
<div id="pic">
<br>
This is a Syncfusion Logo
</div>
<div>
Hello world
</div>
</body>
</html>
```

UWP

```
//Essential PDF supports converting partial webpage to SVG only in Windows
Forms, WPF, ASP.NET, ASP.NET MVC platforms.
```

ASP.NET CORE

```
//Essential PDF supports converting partial webpage to SVG only in Windows
Forms, WPF, ASP.NET, ASP.NET MVC platforms.
```

XAMARIN

```
//Essential PDF supports converting partial webpage to SVG only in Windows
Forms, WPF, ASP.NET, ASP.NET MVC platforms.
```

Working with Optical Character Recognition (OCR)

Essential PDF provides support for Optical Character Recognition with the help of Google's Tesseract Optical Character Recognition engine.

Prerequisites and setting up the Tesseract Engine

- To use the OCR feature in your application, you need to add reference to the following set of assemblies.
 1. Syncfusion.Compression.Base.dll
 2. Syncfusion.Pdf.Base.dll
 3. Syncfusion.OCRProcessor.Base.dll
- Place the SyncfusionTesseract.dll and liblpt168.dll Tesseract assemblies in the local system and provide the assembly path to the OCR processor.

C#

```
OCRProcessor processor = new OCRProcessor(@"TesseractBinaries\")
```

VB.NET

```
Dim processor As New OCRProcessor("TesseractBinaries\")
```

- Place the Tesseract language data {E.g eng.traineddata} in the local system and provide a path to the OCR processor

C#

```
OCRProcessor processor = new OCRProcessor(@"TesseractBinaries\");  
processor.PerformOCR(lDoc, @"TessData\");
```

VB.NET

```
Dim processor As New OCRProcessor("TesseractBinaries\  
processor.PerformOCR(lDoc, "TessData\")
```

You can also download the language packages from below link

<https://github.com/tesseract-ocr/tessdata>

Note: From 16.1.0.24 OCR is not a part of Essential Studio and is available as separate package (OCR Processor) under the Add-On section in the below link <https://www.syncfusion.com/downloads/latest-version>.

Note: PDF supports OCR only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.

Performing OCR for an entire document

You can perform OCR on PDF document with the help of [OCRProcessor](#) Class. Refer the below code snippet for the same.

C#

```
//Initialize the OCR processor by providing the path of tesseract
binaries(SyncfusionTesseract.dll and liblpt168.dll)
using (OCRProcessor processor = new OCRProcessor(@"TesseractBinaries\"))
{
    //Load a PDF document
    PdfLoadedDocument lDoc = new PdfLoadedDocument("Input.pdf");
    //Set OCR language to process
    processor.Settings.Language = Languages.English;
    //Process OCR by providing the PDF document and Tesseract data
    processor.PerformOCR(lDoc, @"TessData\");
    //Save the OCR processed PDF document in the disk
    lDoc.Save("Sample.pdf");
    lDoc.Close(true);
}
```

VB.NET

```
'Initialize the OCR processor by providing the path of tesseract
binaries(SyncfusionTesseract.dll and liblpt168.dll)
Using processor As New OCRProcessor("TesseractBinaries\")
'Load a PDF document
Dim lDoc As New PdfLoadedDocument("Input.pdf")
'Set OCR language to process
processor.Settings.Language = Languages.English
'Process OCR by providing the PDF document and Tesseract data
processor.PerformOCR(lDoc, "TessData\")
'Save the OCR processed PDF document in the disk
lDoc.Save("Sample.pdf")
lDoc.Close(True)
End Using
```

Note: The PerformOCR method returns only the text OCR'd by OCRProcessor. Other existing text in the PDF page won't be returned in this method. Please check [text extraction](#) feature for this.

Performing OCR with tesseract version 3.05

You can perform OCR using the tesseract version 3.05. The [TesseractVersion](#) property is used to switch the tesseract version between 3.02 and 3.05. By default, OCR works with tesseract version 3.02.

You must use the pre built Syncfusion tesseract version 3.05 in the sample to run the OCR properly. The tesseract binaries are shipping with Syncfusion NuGet package, use the following link to download the NuGet package.

<https://www.nuget.org/packages/Syncfusion.OCRProcessor.Base>

The following sample code snippet demonstrates the OCR processor with Tesseract3.05 for PDF documents.

C#

```
using (OCRProcessor processor = new OCRProcessor(@"Tesseract3.05Binaries \"))
{
    //Load a PDF document
    PdfLoadedDocument lDoc = new PdfLoadedDocument("input.pdf");
    //Set OCR language to process
    processor.Settings.Language = Languages.English;
    //Set tesseract OCR Engine
```

```

processor.Settings.TesseractVersion = TesseractVersion.Version3_05;
//Process OCR by providing the PDF document and tesseract data, and enabling
the isMemoryOptimized property
processor.PerformOCR(lDoc, @"TessData\", true);
//Save the OCR processed PDF document in the disk
lDoc.Save("Sample.pdf");
lDoc.Close(true);
}

```

VB.NET

```

Using processor As New OCRProcessor("Tesseract3.05Binaries\")
'Load a PDF document
Dim lDoc As New PdfLoadedDocument("Input.pdf")
'Set OCR language to process
processor.Settings.Language = Languages.English
'Set tesseract OCR engine
processor.Settings.TesseractVersion = TesseractVersion.Version3_05
'Process OCR by providing the PDF document and tesseract data, and enabling
the isMemoryOptimized property
processor.PerformOCR(lDoc, "TessData\", True)
'Save the OCR processed PDF document in the disk
lDoc.Save("Sample.pdf")
lDoc.Close(True)
End Using

```

Performing OCR for a region of the document

You can perform OCR on particular region or several regions of a PDF page with the help of [PageRegion](#) class. Refer the below code snippet for the same.

C#

```

//Initialize the OCR processor by providing the path of the tesseract
binaries(SyncfusionTesseract.dll and liblpt168.dll)
using (OCRProcessor processor = new OCRProcessor(@"TesseractBinaries\"))
{
//Load a PDF document
PdfLoadedDocument lDoc = new PdfLoadedDocument("Input.pdf");
//Set OCR language to process
processor.Settings.Language = Languages.English;
RectangleF rect = new RectangleF(0, 100, 950, 150);
//Assign rectangles to the page
List<PageRegion> pageRegions = new List<PageRegion>();
PageRegion region = new PageRegion();
region.PageIndex = 1;
region.PageRegions = new RectangleF[] { rect };
pageRegions.Add(region);
processor.Settings.Regions = pageRegions;
//Process OCR by providing the PDF document and Tesseract data
processor.PerformOCR(lDoc, @"TessData\"");
//Save the OCR processed PDF document in the disk
lDoc.Save("Sample.pdf");
lDoc.Close(true);
}

```


VB.NET

```

'Initialize the OCR processor by providing the path of the tesseract
binaries(SyncfusionTesseract.dll and liblpt168.dll)
Using processor As New OCRProcessor("TesseractBinaries\")
'Load a PDF document
Dim lDoc As New PdfLoadedDocument("Input.pdf")
'Set OCR language to process
processor.Settings.Language = Languages.English
Dim rect As New RectangleF(0, 100, 950, 150)
'Assign rectangles to the page
Dim pageRegions As New List(Of PageRegion)()
Dim region As New PageRegion()
region.PageIndex = 1
region.PageRegions = New RectangleF() {rect}
pageRegions.Add(region)
processor.Settings.Regions = pageRegions
'Process OCR by providing the PDF document and Tesseract data
processor.PerformOCR(lDoc, "TessData\")
'Save the OCR processed PDF document in the disk
lDoc.Save("Sample.pdf")
lDoc.Close(True)

```

Performing OCR on image

You can perform OCR on an image also. Refer the below code snippets for the same.

C#

```

//Initialize the OCR processor by providing the path of the tesseract
binaries(SyncfusionTesseract.dll and liblpt168.dll)
using (OCRProcessor processor = new OCRProcessor(@"TesseractBinaries\"))
{
    //loading the input image
    Bitmap image = new Bitmap("input.jpeg");
    //Set OCR language to process
    processor.Settings.Language = Languages.English;
    //Process OCR by providing the bitmap image, data dictionary and language
    string ocrText= processor.PerformOCR(image, @"TessData\");
}

```

VB.NET

```

'Initialize the OCR processor by providing the path of the tesseract
binaries(SyncfusionTesseract.dll and liblpt168.dll)
Using processor As New OCRProcessor("TesseractBinaries\")
'loading the input image
Dim image As New Bitmap("input.jpeg")
'Set OCR language to process
processor.Settings.Language = Languages.English
'Process OCR by providing the bitmap image, data dictionary and language
Dim ocrText As String = processor.PerformOCR(image, "TessData\")
End Using

```

Performing OCR for large PDF documents

You can optimize the memory to perform OCR for large PDF documents by enabling the `isMemoryOptimized` property in `PerformOCR` method of `OCRProcessor` class. Optimization will be effective only with Multithreading environment or PDF document with more images. This is demonstrated in the following code sample.

C#

```
//Initialize the OCR processor by providing the path of tesseract
//binaries(SyncfusionTesseract.dll and liblpt168.dll)
using (OCRProcessor processor = new OCRProcessor(@"TesseractBinaries\"))
{
    //Load a PDF document.
    PdfLoadedDocument lDoc = new PdfLoadedDocument("Input.pdf");
    //Set OCR language to process.
    processor.Settings.Language = Languages.English;
    //Process OCR by providing the PDF document, Tesseract data and enable
    //isMemoryOptimized property
    processor.PerformOCR(lDoc, @"TessData\", true);
    //Save the OCR processed PDF document in the disk.
    lDoc.Save("Sample.pdf");
    lDoc.Close(true);
}
```

VB.NET

```
'Initialize the OCR processor by providing the path of tesseract
//binaries(SyncfusionTesseract.dll and liblpt168.dll)
Using processor As New OCRProcessor("TesseractBinaries\")
    'Load a PDF document.
    Dim lDoc As New PdfLoadedDocument("Input.pdf")
    'Set OCR language to process.
    processor.Settings.Language = Languages.English
    'Process OCR by providing the PDF document and Tesseract data enable
    //isMemoryOptimized property.
    processor.PerformOCR(lDoc, "TessData\", True)
    'Save the OCR processed PDF document in the disk.
    lDoc.Save("Sample.pdf")
    lDoc.Close(True)
End Using
```

Performing OCR on rotated page of PDF document

You can perform OCR on the rotated page of a PDF document. Refer to the following code snippet for the same.

C#

```
//Initialize the OCR processor by providing the path of tesseract
//binaries(SyncfusionTesseract.dll and liblpt168.dll)
using (OCRProcessor processor = new OCRProcessor(@"TesseractBinaries\"))
{
    //Load a PDF document
    PdfLoadedDocument lDoc = new PdfLoadedDocument("Input.pdf");
    //Set OCR language to process
    processor.Settings.Language = Languages.English;
```

```
//Set OCR page auto detection rotation
processor.Settings.AutoDetectRotation = true;
//Process OCR by providing the PDF document
processor.PerformOCR(lDoc, @"TessData\");
//Save the OCR processed PDF document in the disk
lDoc.Save("Sample.pdf");
lDoc.Close(true);
}
```

VB.NET

```
'Initialize the OCR processor by providing the path of tesseract
binaries(SyncfusionTesseract.dll and liblpt168.dll)
Using processor As New OCRProcessor("TesseractBinaries\")
'Load a PDF document.
Dim lDoc As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Set OCR language to process
processor.Settings.Language = Languages.English
'Set OCR page auto detection rotation
processor.Settings.AutoDetectRotation = true
'Process OCR by providing the PDF document
processor.PerformOCR(lDoc, "TessData\")
'Save the OCR processed PDF document in the disk
lDoc.Save("Sample.pdf")
lDoc.Close(true)
End Using
```

Layout result from OCR

You can get the OCR'd text and its bounds from a scanned PDF document by using the [OCRLayoutResult](#) Class. Refer to the following code snippet.

C#

```
//Initialize the OCR processor by providing the path of tesseract binaries
(SyncfusionTesseract.dll and liblpt168.dll)
using (OCRProcessor processor = new OCRProcessor(@"TesseractBinaries\"))
{
    //Load a PDF document
    PdfLoadedDocument lDoc = new PdfLoadedDocument("Input.pdf");
    //Set OCR language to process
    processor.Settings.Language = Languages.English;
    //Initializes OCR layout result
    OCRLayoutResult result;
    //Process OCR by providing the PDF document, Tesseract data, and layout
    result
    processor.PerformOCR(lDoc, @"TessData\", out result);
    //Get OCR'd line collection from first page
    OCRLineCollection lines = result.Pages[0].Lines;
    //Get each OCR'd line and its bounds
    foreach (Line line in lines)
    {
        string text = line.Text;
        RectangleF bounds = line.Rectangle;
    }
    //Save the OCR processed PDF document in the disk
}
```

```

lDoc.Save("Sample.pdf");
//Close the document
lDoc.Close(true);
}

```

VB.NET

```

'Initialize the OCR processor by providing the path of tesseract binaries
(SyncfusionTesseract.dll and libleft168.dll)
Using processor As New OCRProcessor("TesseractBinaries\")
'Load a PDF document
Dim lDoc As New PdfLoadedDocument("Input.pdf")
'Set OCR language to process
processor.Settings.Language = Languages.English
'Initializes OCR layout result
Dim result As OCRLayoutResult
'Process OCR by providing the PDF document, Tesseract data, and layout
result
processor.PerformOCR(lDoc, "TessData\ ", result)
'Get OCR'd line collection from first page
Dim lines As OCRLineCollection = result.Pages(0).Lines
'Get each OCR'd line and its bounds
For Each line As Line In lines
Dim text As String = line.Text
Dim bounds As RectangleF = line.Rectangle
Next
'Save the OCR processed PDF document in the disk
lDoc.Save("Sample.pdf")
'Close the document
lDoc.Close(True)
End Using

```

Native call

Enable native call will not launch any temporary process for OCR processing, instead it will invoke the native calls.

Tesseract 3.02

Tesseract 3.02 supports only 32-bit version. By default, this property will be disabled.

Note: Enable native call will not work in 64-bit in Tesseract 3.02 version. Instead a temporary process will be launched for OCR processing.

The following sample code snippet demonstrates the OCR processor with native call support of tesseract 3.02.

C#

```

using (OCRProcessor processor = new OCRProcessor(@"Tesseract3.02Binaries\"))
{
//Load a PDF document
PdfLoadedDocument lDoc = new PdfLoadedDocument("input.pdf");
//Set OCR language to process
processor.Settings.Language = Languages.English;
//Set tesseract OCR Engine
processor.Settings.TesseractVersion = TesseractVersion.Version3_02;
}

```

```
//Process OCR by providing the PDF document and tesseract data, and enabling
the isMemoryOptimized property
processor.PerformOCR(lDoc, @"TessData\", true);
//Save the OCR processed PDF document in the disk
lDoc.Save("Sample.pdf");
lDoc.Close(true);
}
```

VB.NET

```
Using processor As New OCRProcessor("Tesseract3.02Binaries\")
'Load a PDF document
Dim lDoc As New PdfLoadedDocument("Input.pdf")
'Set OCR language to process
processor.Settings.Language = Languages.English
'Set tesseract OCR engine
processor.Settings.TesseractVersion = TesseractVersion.Version3_02
'Process OCR by providing the PDF document and tesseract data, and enabling
the isMemoryOptimized property
processor.PerformOCR(lDoc, "TessData\", True)
'Save the OCR processed PDF document in the disk
lDoc.Save("Sample.pdf")
lDoc.Close(True)
End Using
```

Tesseract 3.05

Tesseract 3.05 supports the native call for both x86 and x64 architectures. By default, the x86 tesseract binaries are available with NuGet package or the tesseract installer.

You can download the x64 supporting tesseract binaries from the following link.

[Tesseract 64-bit binaries](#)

Note: This 64-bit binaries are required only when the native call property is enabled.

Make sure to provide the 64-bit binaries path while using in the 64-bit environment.

The following sample code snippet demonstrates the OCR processor with native call support of tesseract 3.05.

C#

```
using (OCRProcessor processor = new OCRProcessor(@" Tesseract3.05Binaries
\"))
{
//Load a PDF document
PdfLoadedDocument lDoc = new PdfLoadedDocument("input.pdf");
//Set OCR language to process
processor.Settings.Language = Languages.English;
//Set tesseract OCR engine
processor.Settings.TesseractVersion = TesseractVersion.Version3_05;
//Set enable native call
processor.Settings.EnableNativeCall = true;
//Process OCR by providing the PDF document and tesseract data, and enabling
the isMemoryOptimized property
processor.PerformOCR(lDoc, @"TessData\", true);
//Save the OCR processed PDF document in the disk
```

```
lDoc.Save("Sample.pdf");
lDoc.Close(true);
}
```

VB.NET

```
Using processor As New OCRProcessor("Tesseract3.05Binaries\")
    'Load a PDF document
    Dim lDoc As New PdfLoadedDocument("input.pdf")
    'Set OCR language to process
    processor.Settings.Language = Languages.English
    'Set tesseract OCR engine
    processor.Settings.TesseractVersion = TesseractVersion.Version3_05
    'Set enable native call
    processor.Settings.EnableNativeCall = True
    'Process OCR by providing the PDF document and tesseract data, and enabling
    the isMemoryOptimized property
    processor.PerformOCR(lDoc, "TessData\", True)
    'Save the OCR processed PDF document in the disk
    lDoc.Save("Sample.pdf")
    lDoc.Close(True)
End Using
```

Customizing temp folder

While performing OCR on an existing scanned PDF document, the OCR Processor will create temporary files (.temp, .tiff, .txt) and the files are deleted after the process is completed. You can change this temporary files folder location using the [TempFolder](#) property available in the [OCRSettings](#) Instance. Refer to the following code snippet.

C#

```
//Initialize the OCR processor by providing the path of tesseract binaries
(SyncfusionTesseract.dll and libtesseract.dll)
using (OCRProcessor processor = new OCRProcessor(@"TesseractBinaries\"))
{
    //Load a PDF document
    PdfLoadedDocument lDoc = new PdfLoadedDocument("Input.pdf");
    //Set OCR language to process
    processor.Settings.Language = Languages.English;
    //Set custom temp file path location
    processor.Settings.TempFolder = "D:/Temp/";
    //Process OCR by providing the PDF document and Tesseract data
    processor.PerformOCR(lDoc, @"TessData\");
    //Save the OCR processed PDF document in the disk
    lDoc.Save("Sample.pdf");
    //Close the document
    lDoc.Close(true);
}
```

VB.NET

```
'Initialize the OCR processor by providing the path of tesseract binaries
(SyncfusionTesseract.dll and libtesseract.dll)
Using processor As New OCRProcessor("TesseractBinaries\")
```

```
'Load a PDF document
Dim lDoc As New PdfLoadedDocument("Input.pdf")
'Set OCR language to process
processor.Settings.Language = Languages.English
'Set custom temp file path location
processor.Settings.TempFolder = "D:/Temp/"
'Process OCR by providing the PDF document and Tesseract data
processor.PerformOCR(lDoc, "TessData\")
'Save the OCR processed PDF document in the disk
lDoc.Save("Sample.pdf")
'Close the document
lDoc.Close(True)
End Using
```

Best Practices

You can improve the accuracy of the OCR process by choosing the correct compression method when converting the scanned paper to a TIFF image and then to a PDF document.

- Use (zip) lossless compression for color or gray-scale images.
- Use CCITT Group 4 or JBIG2 (lossless) compression for monochrome images. This ensures that optical character recognition works on the highest-quality image, thereby improving the OCR accuracy. This is especially useful in low-resolution scans.
- In addition, rotated images and skewed images can also affect the accuracy and readability of the OCR process.

Tesseract works best with text when at least 300 dots per inch (DPI) are used, so it is beneficial to resize images.

For more details regarding quality improvement, refer to the following link:

<https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality>

You can set the different performance level to the OCRProcessor using [Performance](#) enumeration.

- Rapid – high speed OCR performance and provide normal OCR accuracy
- Fast – provides moderate OCR processing speed and accuracy
- Slow – Slow OCR performance and provide best OCR accuracy.

Refer below code snippet to set the performance of the OCR.

C#

```
OCRProcessor processor = new OCRProcessor(@"TesseractBinaries\")
//set the OCR performance
processor.Settings.Performance = Performance.Fast;
```

VB.NET

```
Dim processor As New OCRProcessor("TesseractBinaries\")
'Set the OCR performance
processor.Settings.Performance = Performance.Fast
```

Troubleshooting

Issue: You can get the exception “Tesseract has not been initialized” while performing OCR process.

Solution 1: To resolve this, make sure the path of the Tesseract binaries and Tesseract data are properly provided as shown below.

C#

```
//TesseractBinaries - path of the folder containing SyncfusionTesseract.dll
and liblpt168.dll
OCRProcessor processor = new OCRProcessor(@"TesseractBinaries\");
//TessData - path of the folder containing the language pack
processor.PerformOCR(lDoc, @"TessData\");
```

VB.NET

```
'TesseractBinaries - path of the folder containing SyncfusionTesseract.dll
and liblpt168.dll
Dim processor As New OCRProcessor("TesseractBinaries\")
'TessData - path of the folder containing the language pack
processor.PerformOCR(lDoc, "TessData\")
```

Solution 2: Make sure that your data file version is 3.02, since the OCR processor is built with Tesseract version 3.02.

Issue: OCR processor doesn’t process languages other than English.

Solution: Essential PDF supports all the languages supported by Tesseract engine.

The dictionary packs for the languages can be downloaded from the following online location:

<https://github.com/tesseract-ocr/tesseract/wiki/Data-Files#data-files-for-version-302>

It is also mandatory to change the corresponding language code in the OCRProcessor.Settings.Language property. For example, to perform optical character recognition in German, the property should be set as processor.Settings.Language = "deu";

The following link contains the complete set of languages supported by Tesseract and their language codes.

<https://github.com/tesseract-ocr/tesseract/blob/master/doc/tesseract.1.asc#languages>

Working with Hyperlinks

In PDF, hyperlinks can be added to allow the users to navigate to another part of PDF file, web page or any other external content. Essential PDF provides support for all these types of hyperlink.

Working with Web navigation

You can navigate to specified URL from a PDF document by using the [PdfTextWebLink](#) class.

Please refer the below code snippet for navigating to the web page.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
```



```

//Create the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12f);
//Create the Text Web Link.
PdfTextWebLink textLink = new PdfTextWebLink();
//Set the hyperlink
textLink.Url = "http://www.syncfusion.com";
//Set the link text
textLink.Text = "Syncfusion .NET components and controls";
//Set the font
textLink.Font = font;
//Draw the hyperlink in PDF page
textLink.DrawTextWebLink(page, new PointF(10, 40));
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 12.0F)
'Create the Text Web Link.
Dim textLink As New PdfTextWebLink()
'Add the hyperlink
textLink.Url = "http://www.syncfusion.com"
'Set the link text
textLink.Text = "Syncfusion .NET components and controls"
'Set the font
textLink.Font = font
'Draw the hyperlink in PDF page
textLink.DrawTextWebLink(loadedPage.Graphics, New PointF(10, 40))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12f);
//Create the Text Web Link.
PdfTextWebLink textLink = new PdfTextWebLink();
//Set the hyperlink
textLink.Url = "http://www.syncfusion.com";
//Set the link text
textLink.Text = "Syncfusion .NET components and controls";
//Set the font
textLink.Font = font;

```

```
//Draw the hyperlink in PDF page
textLink.DrawTextWebLink(page, new PointF(10, 40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12f);
//Create the Text Web Link.
PdfTextWebLink textLink = new PdfTextWebLink();
//Set the hyperlink
textLink.Url = "http://www.syncfusion.com";
//Set the link text
textLink.Text = "Syncfusion .NET components and controls";
//Set the font
textLink.Font = font;
//Draw the hyperlink in PDF page
textLink.DrawTextWebLink(page, new PointF(10, 40));
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12f);
//Create the Text Web Link.
PdfTextWebLink textLink = new PdfTextWebLink();
//Set the hyperlink
textLink.Url = "http://www.syncfusion.com";
//Set the link text
```

```

textLink.Text = "Syncfusion .NET components and controls";
//Set the font
textLink.Font = font;
//Draw the hyperlink in PDF page
textLink.DrawTextWebLink(page, new PointF(10, 40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

To add a web hyperlink to an existing document, please refer the below code snippet.

C#

```

//Load the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(@"Filename.pdf");
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12f);
//Create the Text Web Link.
PdfTextWebLink textLink = new PdfTextWebLink();
//Set the hyperlink
textLink.Url = "http://www.syncfusion.com";
//Set the link text
textLink.Text = "Syncfusion .NET components and controls";
//Set the font
textLink.Font = font;
//Draw the hyperlink in loaded page graphics
textLink.DrawTextWebLink(loadedPage.Graphics, new PointF(10, 40));
//Save the document.
loadedDocument.Save("Output.pdf");
//Close the document.
loadedDocument.Close(true);

```

VB.NET

```

'Load the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument("fileName.pdf")
'Load the page

```

```

Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Create the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 12.0F)
'Create the Text Web Link.
Dim textLink As New PdfTextWebLink()
'Add the hyperlink
textLink.Url = "http://www.syncfusion.com"
'Set the link text
textLink.Text = "Syncfusion .NET components and controls"
'Set the font
textLink.Font = font
'Draw the hyperlink in loaded page graphics
textLink.DrawTextWebLink(loadedPage.Graphics, New PointF(10, 40))
'Save the document.
loadedDocument.Save("Output1.pdf")
'Close the document.
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12f);
//Create the Text Web Link.
PdfTextWebLink textLink = new PdfTextWebLink();
//Set the hyperlink
textLink.Url = "http://www.syncfusion.com";
//Set the link text
textLink.Text = "Syncfusion .NET components and controls";
//Set the font
textLink.Font = font;
//Draw the hyperlink in loaded page graphics
textLink.DrawTextWebLink(loadedPage.Graphics, new PointF(10, 40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12f);
//Create the Text Web Link.
PdfTextWebLink textLink = new PdfTextWebLink();
//Set the hyperlink
textLink.Url = "http://www.syncfusion.com";
//Set the link text
textLink.Text = "Syncfusion .NET components and controls";
//Set the font
textLink.Font = font;
//Draw the hyperlink in loaded page graphics
textLink.DrawTextWebLink(loadedPage.Graphics, new PointF(10, 40));
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Closes the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12f);
//Create the Text Web Link.
PdfTextWebLink textLink = new PdfTextWebLink();
//Set the hyperlink
textLink.Url = "http://www.syncfusion.com";
//Set the link text
textLink.Text = "Syncfusion .NET components and controls";
//Set the font
textLink.Font = font;
//Draw the hyperlink in loaded page graphics
textLink.DrawTextWebLink(loadedPage.Graphics, new PointF(10, 40));
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
```

```
//Closes the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

Working with internal document navigation

To allow the users to navigate to any other part of the same document, [PdfDocumentLinkAnnotation](#) class can be used. The below code explains how to add the hyperlink for internal document navigation.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Create a new page
PdfPage page = document.Pages.Add();
//Create a new rectangle
RectangleF docLinkAnnotationBounds = new RectangleF(10, 40, 30, 30);
//Create a new document link annotation.
PdfDocumentLinkAnnotation documentLinkAnnotation = new
PdfDocumentLinkAnnotation(docLinkAnnotationBounds);
//Set the annotation flags.
documentLinkAnnotation.AnnotationFlags = PdfAnnotationFlags.NoRotate;
//Set the annotation text.
documentLinkAnnotation.Text = "Document link annotation";
//Set the annotation's color.
documentLinkAnnotation.Color = new PdfColor(Color.Navy);
//Creates another page
PdfPage navigationPage = document.Pages.Add();
//Set the destination.
documentLinkAnnotation.Destination = new PdfDestination(navigationPage);
//Set the document link annotation location.
documentLinkAnnotation.Destination.Location = new Point(10, 0);
//Set the document annotation zoom level
documentLinkAnnotation.Destination.Zoom = 5;
//Add this annotation to a new page.
page.Annotations.Add(documentLinkAnnotation);
//Save the document to disk.
document.Save("DocumentLinkAnnotation.pdf");
//Close the document
document.Close();
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Create a new page
Dim page As PdfPage = document.Pages.Add()
'Create a new rectangle
Dim docLinkAnnotationBounds As New RectangleF(10, 40, 30, 30)
'Create a new document link annotation.
```

```

Dim documentLinkAnnotation As New
PdfDocumentLinkAnnotation(docLinkAnnotationBounds)
'Set the annotation flags.
documentLinkAnnotation.AnnotationFlags = PdfAnnotationFlags.NoRotate
'Set the annotation text.
documentLinkAnnotation.Text = "Document link annotation"
'Set the annotation's color.
documentLinkAnnotation.Color = New PdfColor(Color.Navy)
'Create another page
Dim navigationPage As PdfPage = document.Pages.Add()
'Set the destination.
documentLinkAnnotation.Destination = New PdfDestination(navigationPage)
'Set the document link annotation location.
documentLinkAnnotation.Destination.Location = New Point(10, 0)
'Set the document annotation zoom level
documentLinkAnnotation.Destination.Zoom = 5
'Add this annotation to a new page.
page.Annotations.Add(documentLinkAnnotation)
'Save the document to disk.
document.Save("DocumentLinkAnnotation.pdf")
'Close the document
document.Close()

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Create a new page
PdfPage page = document.Pages.Add();
//Create a new rectangle
RectangleF docLinkAnnotationBounds = new RectangleF(10, 40, 30, 30);
//Create a new document link annotation.
PdfDocumentLinkAnnotation documentLinkAnnotation = new
PdfDocumentLinkAnnotation(docLinkAnnotationBounds);
//Set the annotation flags.
documentLinkAnnotation.AnnotationFlags = PdfAnnotationFlags.NoRotate;
//Set the annotation text.
documentLinkAnnotation.Text = "Document link annotation";
//Set the annotation's color.
documentLinkAnnotation.Color = new PdfColor(System.Drawing.Color.FromArgb(0,
0, 0, 128));
//Creates another page
PdfPage navigationPage = document.Pages.Add();
//Set the destination.
documentLinkAnnotation.Destination = new PdfDestination(navigationPage);
//Set the document link annotation location.
documentLinkAnnotation.Destination.Location = new PointF(10, 0);
//Set the document annotation zoom level
documentLinkAnnotation.Destination.Zoom = 5;
//Add this annotation to a new page.
page.Annotations.Add(documentLinkAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);

```

```
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "DocumentLinkAnnotation.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Create a new page
PdfPage page = document.Pages.Add();
//Create a new rectangle
RectangleF docLinkAnnotationBounds = new RectangleF(10, 40, 30, 30);
//Create a new document link annotation.
PdfDocumentLinkAnnotation documentLinkAnnotation = new
PdfDocumentLinkAnnotation(docLinkAnnotationBounds);
//Set the annotation flags.
documentLinkAnnotation.AnnotationFlags = PdfAnnotationFlags.NoRotate;
//Set the annotation text.
documentLinkAnnotation.Text = "Document link annotation";
//Set the annotation's color.
documentLinkAnnotation.Color = new PdfColor(Color.Navy);
//Creates another page
PdfPage navigationPage = document.Pages.Add();
//Set the destination.
documentLinkAnnotation.Destination = new PdfDestination(navigationPage);
//Set the document link annotation location.
documentLinkAnnotation.Destination.Location = new PointF(10, 0);
//Set the document annotation zoom level
documentLinkAnnotation.Destination.Zoom = 5;
//Add this annotation to a new page.
page.Annotations.Add(documentLinkAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "DocumentLinkAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Create a new page
PdfPage page = document.Pages.Add();
//Create a new rectangle
RectangleF docLinkAnnotationBounds = new RectangleF(10, 40, 30, 30);
//Create a new document link annotation.
PdfDocumentLinkAnnotation documentLinkAnnotation = new
PdfDocumentLinkAnnotation(docLinkAnnotationBounds);
```



```

//Set the annotation flags.
documentLinkAnnotation.AnnotationFlags = PdfAnnotationFlags.NoRotate;
//Set the annotation text.
documentLinkAnnotation.Text = "Document link annotation";
//Set the annotation's color.
documentLinkAnnotation.Color = new PdfColor(Syncfusion.Drawing.Color.Navy);
//Creates another page
PdfPage navigationPage = document.Pages.Add();
//Set the destination.
documentLinkAnnotation.Destination = new PdfDestination(navigationPage);
//Set the document link annotation location.
documentLinkAnnotation.Destination.Location = new PointF(10, 0);
//Set the document annotation zoom level
documentLinkAnnotation.Destination.Zoom = 5;
//Add this annotation to a new page.
page.Annotations.Add(documentLinkAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DocumentLinkA
nnotation.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("DocumentLinkAnnotation.pd
f", "application/pdf", stream);
}

```

To add a [PdfDocumentLinkAnnotation](#) to an existing document, please use the below code snippet.

C#

```

//Load the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(@"fileName.pdf");
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a new rectangle
RectangleF docLinkAnnotationBounds = new RectangleF(10, 40, 30, 30);
//Create a new document link annotation.
PdfDocumentLinkAnnotation documentLinkAnnotation = new
PdfDocumentLinkAnnotation(docLinkAnnotationBounds);
//Set the annotation text.
documentLinkAnnotation.Text = "Document link annotation";
//Set the existing page for navigation
PdfLoadedPage navigationPage = loadedDocument.Pages[1] as PdfLoadedPage;
//Set the pdf destination.
documentLinkAnnotation.Destination = new PdfDestination(navigationPage);

```

```
//Set the document link annotation location.
documentLinkAnnotation.Destination.Location = new Point(10, 0);
//Add this annotation to respective page.
loadedPage.Annotations.Add(documentLinkAnnotation);
//Save the document to disk.
loadedDocument.Save("DocumentLinkAnnotation.pdf");
//Close the document
loadedDocument.Close();
```

VB.NET

```
'Load the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument("fileName.pdf")
'Load the page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Create a new rectangle
Dim docLinkAnnotationBounds As New RectangleF(10, 40, 30, 30)
'Create a new document link annotation.
Dim documentLinkAnnotation As New
PdfDocumentLinkAnnotation(docLinkAnnotationBounds)
'Set the annotation text.
documentLinkAnnotation.Text = "Document link annotation"
'Set the existing page for navigation
Dim navigationPage As PdfLoadedPage = TryCast(loadedDocument.Pages(1),
PdfLoadedPage)
'Set the pdf destination.
documentLinkAnnotation.Destination = New PdfDestination(navigationPage)
'Set the document link annotation location.
documentLinkAnnotation.Destination.Location = New Point(10, 0)
'Add this annotation to respective page.
loadedPage.Annotations.Add(documentLinkAnnotation)
'Save the document to disk.
loadedDocument.Save("DocumentLinkAnnotation.pdf")
'Close the document
loadedDocument.Close()
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a new rectangle
RectangleF docLinkAnnotationBounds = new RectangleF(10, 40, 30, 30);
//Create a new document link annotation.
PdfDocumentLinkAnnotation documentLinkAnnotation = new
PdfDocumentLinkAnnotation(docLinkAnnotationBounds);
```

```

//Set the annotation text.
documentLinkAnnotation.Text = "Document link annotation";
//Set the existing page for navigation
PdfLoadedPage navigationPage = loadedDocument.Pages[1] as PdfLoadedPage;
//Set the pdf destination.
documentLinkAnnotation.Destination = new PdfDestination(navigationPage);
//Set the document link annotation location.
documentLinkAnnotation.Destination.Location = new System.Drawing.PointF(10,
0);
//Add this annotation to respective page.
loadedPage.Annotations.Add(documentLinkAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "DocumentLinkAnnotation.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("fileName.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a new rectangle
RectangleF docLinkAnnotationBounds = new RectangleF(10, 40, 30, 30);
//Create a new document link annotation.
PdfDocumentLinkAnnotation documentLinkAnnotation = new
PdfDocumentLinkAnnotation(docLinkAnnotationBounds);
//Set the annotation text.
documentLinkAnnotation.Text = "Document link annotation";
//Set the existing page for navigation
PdfLoadedPage navigationPage = loadedDocument.Pages[1] as PdfLoadedPage;
//Set the pdf destination.
documentLinkAnnotation.Destination = new PdfDestination(navigationPage);
//Set the document link annotation location.
documentLinkAnnotation.Destination.Location = new PointF(10, 0);
//Add this annotation to respective page.
loadedPage.Annotations.Add(documentLinkAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Closes the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "DocumentLinkAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a new rectangle
RectangleF docLinkAnnotationBounds = new RectangleF(10, 40, 30, 30);
//Create a new document link annotation.
PdfDocumentLinkAnnotation documentLinkAnnotation = new
PdfDocumentLinkAnnotation(docLinkAnnotationBounds);
//Set the annotation text.
documentLinkAnnotation.Text = "Document link annotation";
//Set the existing page for navigation
PdfLoadedPage navigationPage = loadedDocument.Pages[1] as PdfLoadedPage;
//Set the pdf destination.
documentLinkAnnotation.Destination = new PdfDestination(navigationPage);
//Set the document link annotation location.
documentLinkAnnotation.Destination.Location = new
Syncfusion.Drawing.PointF(10, 0);
//Add this annotation to respective page.
loadedPage.Annotations.Add(documentLinkAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Closes the document
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DocumentLinkA
nnotation.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("DocumentLinkAnnotation.pd
f", "application/pdf", stream);
}

```

Working with external document navigation

You can open external documents like images, text files, PDF, etc. using [PdfFileLinkAnnotation](#) class.

Please refer the below code snippet for navigating to external documents:

C#

```

//Create the PDF document
PdfDocument document = new PdfDocument();

```

```
//Creates a new page
PdfPage page = document.Pages.Add();
//Create a new rectangle
RectangleF bounds = new RectangleF(10, 40, 30, 30);
//Create a link for image file
PdfFileLinkAnnotation fileLinkAnnotation = new PdfFileLinkAnnotation(bounds,
@"..\..\Data\logo.png");
//Add this annotation to a page.
page.Annotations.Add(fileLinkAnnotation);
//Save the document to disk.
document.Save("FileLinkAnnotation.pdf");
//close the document
document.Close();
```

VB.NET

```
'Create the PDF document
Dim document As New PdfDocument()
'Creates a new page and adds it as the last page of the document
Dim page As PdfPage = document.Pages.Add()
'Create a new rectangle
Dim rectangle As New RectangleF(10, 40, 30, 30)
'Create a link for image file
Dim fileLinkAnnotation As New PdfFileLinkAnnotation(rectangle,
"..\..\Data\logo.png")
'Add this annotation to a page.
page.Annotations.Add(fileLinkAnnotation)
'Save the document to disk.
document.Save("FileLinkAnnotation.pdf")
'close the document
document.Close()
```

UWP

```
//PDF supports external document navigation only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//PDF supports external document navigation only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.
```

XAMARIN

```
//PDF supports external document navigation only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.
```

Note: The above link makes use of the absolute path of the file for navigation. So, moving the files to another machine or location may lead to file not found error in PDF reader applications.

To open a file in relative path, the [PdfLaunchAction](#) can be used. While using the relative path in launch action, the files can be moved to any machine, provided the relative path is being maintained. The below code snippet explains the same.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Create a new page
PdfPage page = document.Pages.Add();
//Create the button field
PdfButtonField submitButton = new PdfButtonField(page, "submitButton");
submitButton.Bounds = new RectangleF(100, 160, 100, 20);
submitButton.Text = "Launch";
//Create a the PdfLaunchAction
PdfLaunchAction launchAction = new PdfLaunchAction(@"..\..\Data\
Document.txt", PdfFilePathType.Relative);
//Set the launchAction to the submitButton
submitButton.Actions.GotFocus = launchAction;
//Add the form field
document.Form.Fields.Add(submitButton);
//Save the document to disk.
document.Save("output.pdf");
//close the document
document.Close();
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Create a new page
Dim page As PdfPage = document.Pages.Add()
'Create a new PdfButtonField
Dim submitButton As New PdfButtonField(page, "submitButton")
submitButton.Bounds = New RectangleF(25, 160, 100, 20)
submitButton.Text = "Launch"
'Create a the PdfLaunchAction
Dim launchAction As New PdfLaunchAction(@"..\..\Data\Document.txt",
PdfFilePathType.Relative)
'Set the launch Action to the submitButton
submitButton.Actions.GotFocus = launchAction
'Add the form field
document.Form.Fields.Add(submitButton)
'Save the document to disk.
document.Save("output.pdf")
'close the document
document.Close()
```

UWP

```
//PDF supports external document navigation only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//PDF supports external document navigation only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.
```

XAMARIN

```
//PDF supports external document navigation only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.
```

Working with PDF Templates

A PDF template is a drawing surface, where contents can be added. All the elements that can be added to a [PdfPage](#) is supported in [PdfTemplate](#) as well. The template in turn can be drawn over the page or can be positioned at any part of the page.

Creating a new PDF template

The [PdfTemplate](#) class can be used to create a new PDF template. You can add contents to the template using [Graphics](#) property of the PdfTemplate object.

The below code snippet illustrates how to add contents to the PdfTemplate and render into the new PDF page.

C#

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a PDF Template.
PdfTemplate template = new PdfTemplate(100, 50);
//Draw a rectangle on the template graphics
template.Graphics.DrawRectangle(PdfBrushes.BurlyWood, new
System.Drawing.RectangleF(0, 0, 100, 50));
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
PdfBrush brush = new PdfSolidBrush(Color.Black);
//Draw a string using the graphics of the template.
template.Graphics.DrawString("Hello World", font, brush, 5, 5);
//Draw the template on the page graphics of the document.
pdfPage.Graphics.DrawPdfTemplate(template, PointF.Empty);
//Save the document.
pdfDocument.Save("Output.pdf");
//close the document
pdfDocument.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim pdfDocument As New PdfDocument()
'Add a page to the PDF document.
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
'Create a PDF Template.
Dim template As New PdfTemplate(100, 50)
'Draw a rectangle on the template graphics.
template.Graphics.DrawRectangle(PdfBrushes.BurlyWood, New
System.Drawing.RectangleF(0, 0, 100, 50))
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 14)
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
'Draw a string using the graphics of the template.
template.Graphics.DrawString("Hello World", font, brush, 5, 5)
'Draw the template on the page graphics of the document.
```

```
pdfPage.Graphics.DrawPdfTemplate(template, PointF.Empty)
'Save the document.
pdfDocument.Save("Output.pdf")
'close the document
pdfDocument.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a PDF Template.
PdfTemplate template = new PdfTemplate(100, 50);
//Draw a rectangle on the template graphics
template.Graphics.DrawRectangle(PdfBrushes.BurlyWood, new
System.Drawing.RectangleF(0, 0, 100, 50));
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
PdfBrush brush = new PdfSolidBrush(Color.FromArgb(255, 0, 0, 0));
//Draw a string using the graphics of the template.
template.Graphics.DrawString("Hello World", font, brush, 5, 5);
//Draw the template on the page graphics of the document.
pdfPage.Graphics.DrawPdfTemplate(template, PointF.Empty);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
//Close the document
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a PDF Template.
PdfTemplate template = new PdfTemplate(100, 50);
//Draw a rectangle on the template graphics
template.Graphics.DrawRectangle(PdfBrushes.BurlyWood, new
Syncfusion.Drawing.RectangleF(0, 0, 100, 50));
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
PdfBrush brush = new PdfSolidBrush(Color.Black);
//Draw a string using the graphics of the template.
template.Graphics.DrawString("Hello World", font, brush, 5, 5);
//Draw the template on the page graphics of the document.
pdfPage.Graphics.DrawPdfTemplate(template, PointF.Empty);
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Closes the document
pdfDocument.Close(true);
//Defining the ContentType for pdf file
```



```

string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a PDF Template.
PdfTemplate template = new PdfTemplate(100, 50);
//Draw a rectangle on the template graphics
template.Graphics.DrawRectangle(PdfBrushes.BurlyWood, new
Syncfusion.Drawing.RectangleF(0, 0, 100, 50));
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Black);
//Draw a string using the graphics of the template.
template.Graphics.DrawString("Hello World", font, brush, 5, 5);
//Draw the template on the page graphics of the document.
pdfPage.Graphics.DrawPdfTemplate(template, PointF.Empty);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Closes the document
pdfDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The below code snippet illustrates how to render the [PdfTemplate](#) on an existing PDF document.

C#

```

//Load the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Load the page into Pdf document.
PdfLoadedPage LoadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a PDF Template.
PdfTemplate template = new PdfTemplate(100, 50);
//Draw a rectangle on the template graphics.

```

```

template.Graphics.DrawRectangle(PdfBrushes.BurlyWood, new
System.Drawing.RectangleF(0, 0, 100, 50));
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
PdfBrush brush = new PdfSolidBrush(Color.Black);
//Draw a string on the graphics of the template.
template.Graphics.DrawString("Hello World", font, brush, 5, 5);
//Draw the template on the page graphics of the document.
LoadedPage.Graphics.DrawPdfTemplate(template, PointF.Empty);
//Save the modified document.
loadedDocument.Save("Output.pdf");
//close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Load the page into Pdf document.
Dim LoadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Create a PDF Template.
Dim template As New PdfTemplate(100, 50)
'Draw a rectangle on the template graphics.
template.Graphics.DrawRectangle(PdfBrushes.BurlyWood, New
System.Drawing.RectangleF(0, 0, 100, 50))
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 14)
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
'Draw a string on the graphics of the template.
template.Graphics.DrawString("Hello World", font, brush, 5, 5)
'Draw the template on the page graphics of the document.
LoadedPage.Graphics.DrawPdfTemplate(template, PointF.Empty)
'Save the modified document.
loadedDocument.Save("Output.pdf")
'close the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Load the page into Pdf document.
PdfLoadedPage LoadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a PDF Template.
PdfTemplate template = new PdfTemplate(100, 50);
//Draw a rectangle on the template graphics.
template.Graphics.DrawRectangle(PdfBrushes.BurlyWood, new
System.Drawing.RectangleF(0, 0, 100, 50));
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);

```

```

PdfBrush brush = new PdfSolidBrush(Color.FromArgb(255, 0, 0, 0));
//Draw a string on the graphics of the template.
template.Graphics.DrawString("Hello World", font, brush, 5, 5);
//Draw the template on the page graphics of the document.
LoadedPage.Graphics.DrawPdfTemplate(template, PointF.Empty);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the page into Pdf document.
PdfLoadedPage LoadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a PDF Template.
PdfTemplate template = new PdfTemplate(100, 50);
//Draw a rectangle on the template graphics.
template.Graphics.DrawRectangle(PdfBrushes.BurlyWood, new
Syncfusion.Drawing.RectangleF(0, 0, 100, 50));
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Black);
//Draw a string on the graphics of the template.
template.Graphics.DrawString("Hello World", font, brush, 5, 5);
//Draw the template on the page graphics of the document.
LoadedPage.Graphics.DrawPdfTemplate(template,
Syncfusion.Drawing.PointF.Empty);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Closes the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the page into Pdf document.

```

```

PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a PDF Template.
PdfTemplate template = new PdfTemplate(100, 50);
//Draw a rectangle on the template graphics.
template.Graphics.DrawRectangle(PdfBrushes.BurlyWood, new
Syncfusion.Drawing.RectangleF(0, 0, 100, 50));
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Black);
//Draw a string on the graphics of the template.
template.Graphics.DrawString("Hello World", font, brush, 5, 5);
//Draw the template on the page graphics of the document.
loadedPage.Graphics.DrawPdfTemplate(template, PointF.Empty);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Closes the document
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Creating templates from existing PDF document

Essential PDF supports to create the templates from an existing PDF document page and draw it in on a new PDF document.

The below code illustrates how to create the template from an existing page and draw it in new PDF document.

C#

```

//Load the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create the template from the page.
PdfTemplate template = loadedPage.CreateTemplate();
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Set the document margin
document.PageSettings.SetMargins(2);
//Add the page
PdfPage page = document.Pages.Add();
//Create the graphics
PdfGraphics graphics = page.Graphics;

```

```
//Draw the template
graphics.DrawPdfTemplate(template, PointF.Empty, new
SizeF(page.Size.Width/2, page.Size.Height));
//Save the new document.
document.Save("output.pdf");
//Close the documents
loadedDocument.Close(true);
document.Close(true);
```

VB.NET

```
'Load the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Load the page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Create the template from the page.
Dim template As PdfTemplate = loadedPage.CreateTemplate()
'Create a new PDF document
Dim document As New PdfDocument()
'Set the document margin
document.PageSettings.SetMargins(2)
'Add the page
Dim page As PdfPage = document.Pages.Add()
'Create the graphics
Dim graphics As PdfGraphics = page.Graphics
'Draw the template
graphics.DrawPdfTemplate(template, PointF.Empty, New SizeF(page.Size.Width \
2, page.Size.Height))
'Save the new document.
document.Save("output.pdf")
'Close the documents
loadedDocument.Close(True)
document.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create the template from the page.
PdfTemplate template = loadedPage.CreateTemplate();
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Set the document margin
document.PageSettings.SetMargins(2);
//Add the page
```

```

PdfPage page = document.Pages.Add();
//Create the graphics
PdfGraphics graphics = page.Graphics;
//Draw the template
graphics.DrawPdfTemplate(template, PointF.Empty, new SizeF(page.Size.Width /
2, page.Size.Height));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create the template from the page.
PdfTemplate template = loadedPage.CreateTemplate();
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Set the document margin
document.PageSettings.SetMargins(2);
//Add the page
PdfPage page = document.Pages.Add();
//Create the graphics
PdfGraphics graphics = page.Graphics;
//Draw the template
graphics.DrawPdfTemplate(template, Syncfusion.Drawing.PointF.Empty, new
Syncfusion.Drawing.SizeF(page.Size.Width / 2, page.Size.Height));
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Closes the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");

```

```

PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create the template from the page.
PdfTemplate template = loadedPage.CreateTemplate();
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Set the document margin
document.PageSettings.SetMargins(2);
//Add the page
PdfPage page = document.Pages.Add();
//Create the graphics
PdfGraphics graphics = page.Graphics;
//Draw the template
graphics.DrawPdfTemplate(template, PointF.Empty, new SizeF(page.Size.Width /
2, page.Size.Height));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Closes the document
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
"application/pdf", stream);
}

```

Working with PdfPageTemplateElement

[PdfPageTemplateElement](#) is template element that can be added to any part of the PDF page such as header, footer etc.

The below code illustrates how to add the page template elements in a PDF document.

C#

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a header and draw the image.
RectangleF bounds = new RectangleF(0, 0,
pdfDocument.Pages[0].GetClientSize().Width, 50);
PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
PdfImage image = new PdfBitmap(@"Logo.png");
//Draw the image in the header.
header.Graphics.DrawImage(image, new PointF(0, 0), new SizeF(100, 50));

```

```

//Add the header at the top.
pdfDocument.Template.Top = header;
//Create a Page template that can be used as footer.
PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds );
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 7);
PdfBrush brush = new PdfSolidBrush(Color.Black);
//Create page number field.
PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
//Create page count field.
PdfPageCountField count = new PdfPageCountField(font, brush);
//Add the fields in composite fields.
PdfCompositeField compositeField = new PdfCompositeField(font, brush, "Page
{0} of {1}", pageNumber, count);
compositeField.Bounds = footer.Bounds;
//Draw the composite field in footer.
compositeField.Draw(footer.Graphics, new PointF(470, 40));
//Add the footer template at the bottom.
pdfDocument.Template.Bottom = footer;
//Save and close the document.
pdfDocument.Save("Output.pdf");
pdfDocument.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim pdfDocument As New PdfDocument()
'Add a page to the PDF document.
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
'Create a header and draw the image.
Dim bounds As New RectangleF(0, 0,
pdfDocument.Pages(0).GetClientSize().Width, 50)
Dim header As New PdfPageTemplateElement(bounds )
Dim image As PdfImage = New PdfBitmap("Logo.jpg")
'Draw the image in the Header.
header.Graphics.DrawImage(image, New PointF(0, 0), New SizeF(100, 50))
'Add the header at the top.
pdfDocument.Template.Top = header
'Create a page template that can be used as footer.
Dim footer As New PdfPageTemplateElement(bounds )
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 7)
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
'Create page number field.
Dim pageNumber As New PdfPageNumberField(font, brush)
'Create page count field.
Dim count As New PdfPageCountField(font, brush)
'Add the fields in composite fields.
Dim compositeField As New PdfCompositeField(font, brush, "Page {0} of {1}",
pageNumber, count)
compositeField.Bounds = footer.Bounds
'Draw the composite field in footer.
compositeField.Draw(footer.Graphics, New PointF(470, 40))
'Add the footer template at the bottom.
pdfDocument.Template.Bottom = footer
'Save and close the document.
pdfDocument.Save("Output.pdf")
pdfDocument.Close(True)

```


UWP

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a header and draw the image.
RectangleF bounds = new RectangleF(0, 0,
pdfDocument.Pages[0].GetClientSize().Width, 50);
PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
//Load the image as stream
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.L
PdfImage image = new PdfBitmap(imageStream);
//Draw the image in the header.
header.Graphics.DrawImage(image, new PointF(0, 0), new SizeF(100, 50));
//Add the header at the top.
pdfDocument.Template.Top = header;
//Create a Page template that can be used as footer.
PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds);
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 7);
PdfBrush brush = new PdfSolidBrush(Color.FromArgb(128, 0, 0, 0));
//Create page number field.
PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
//Create page count field.
PdfPageCountField count = new PdfPageCountField(font, brush);
//Add the fields in composite fields.
PdfCompositeField compositeField = new PdfCompositeField(font, brush, "Page
{0} of {1}", pageNumber, count);
compositeField.Bounds = footer.Bounds;
//Draw the composite field in footer.
compositeField.Draw(footer.Graphics, new PointF(470, 40));
//Add the footer template at the bottom.
pdfDocument.Template.Bottom = footer;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
//Close the document
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a header and draw the image.
RectangleF bounds = new RectangleF(0, 0,
pdfDocument.Pages[0].GetClientSize().Width, 50);
PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
//Load the image file

```

```

FileStream imageStream = new FileStream("Logo.png", FileMode.Open,
    FileAccess.Read);
PdfImage image = new PdfBitmap(imageStream);
//Draw the image in the header.
header.Graphics.DrawImage(image, new Syncfusion.Drawing.PointF(0, 0), new
    SizeF(100, 50));
//Add the header at the top.
pdfDocument.Template.Top = header;
//Create a Page template that can be used as footer.
PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds);
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 7);
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Black);
//Create page number field.
PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
//Create page count field.
PdfPageCountField count = new PdfPageCountField(font, brush);
//Add the fields in composite fields.
PdfCompositeField compositeField = new PdfCompositeField(font, brush, "Page
    {0} of {1}", pageNumber, count);
compositeField.Bounds = footer.Bounds;
//Draw the composite field in footer.
compositeField.Draw(footer.Graphics, new Syncfusion.Drawing.PointF(470,
    40));
//Add the footer template at the bottom.
pdfDocument.Template.Bottom = footer;
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Closes the document
pdfDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
    type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a header and draw the image.
RectangleF bounds = new RectangleF(0, 0,
    pdfDocument.Pages[0].GetClientSize().Width, 50);
PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
//Load the image as stream
Stream imageStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
    Logo.png");
PdfImage image = new PdfBitmap(imageStream);
//Draw the image in the header.
header.Graphics.DrawImage(image, new PointF(0, 0), new SizeF(100, 50));

```

```

//Add the header at the top.
pdfDocument.Template.Top = header;
//Create a Page template that can be used as footer.
PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds);
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 7);
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Black);
//Create page number field.
PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
//Create page count field.
PdfPageCountField count = new PdfPageCountField(font, brush);
//Add the fields in composite fields.
PdfCompositeField compositeField = new PdfCompositeField(font, brush, "Page
{0} of {1}", pageNumber, count);
compositeField.Bounds = footer.Bounds;
//Draw the composite field in footer.
compositeField.Draw(footer.Graphics, new PointF(470, 40));
//Add the footer template at the bottom.
pdfDocument.Template.Bottom = footer;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Closes the document
pdfDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Creating Document Overlays

Multiple templates can be drawn over a PDF page, to create a document-overlay. The below code illustrates how to overlay the documents.

C#

```

//Load the existing documents
PdfLoadedDocument loadedDocument1 = new PdfLoadedDocument(fileName1);
PdfLoadedDocument loadedDocument2 = new PdfLoadedDocument(fileName2);
//Create the new document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
//Create a template from the first document
PdfPageBase loadedPage = loadedDocument1.Pages[0];
PdfTemplate template = loadedPage.CreateTemplate();
//Draw the loaded template into new document.

```

```

page.Graphics.DrawPdfTemplate(template, new PointF(0, 0), new SizeF(500,
700));
// Create a template from the second document
loadedPage = loadedDocument2.Pages[0];
template = loadedPage.CreateTemplate();
//Draw the loaded template into new document.
page.Graphics.DrawPdfTemplate(template, new PointF(10, 10), new SizeF(400,
500));
//Save the new document.
document.Save("output.pdf");
//Close the documents
loadedDocument1.Close(true);
loadedDocument2.Close(true);
document.Close(true);

```

VB.NET

```

'Load the existing documents
Dim loadedDocument1 As New PdfLoadedDocument(fileName1)
Dim loadedDocument2 As New PdfLoadedDocument(fileName2)
'Create the new document
Dim document As New PdfDocument()
Dim page As PdfPage = document.Pages.Add()
'Create a template from the first document
Dim loadedPage As PdfPageBase = loadedDocument1.Pages(0)
Dim template As PdfTemplate = loadedPage.CreateTemplate()
'Draw the loaded template into new document.
page.Graphics.DrawPdfTemplate(template, New PointF(0, 0), New SizeF(500,
700))
'Create a template from the first document
loadedPage = loadedDocument2.Pages(0)
template = loadedPage.CreateTemplate()
'Draw the loaded template into new document.
page.Graphics.DrawPdfTemplate(template, New PointF(10, 10), New SizeF(400,
500))
'Save the new document.
document.Save("output.pdf")
'Close the documents
loadedDocument1.Close(True)
loadedDocument2.Close(True)
document.Close(True)

```

UWP

```

//Load the existing documents.
Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Sample1.pdf");
PdfLoadedDocument loadedDocument1 = new PdfLoadedDocument();
await loadedDocument1.OpenAsync(pdfStream);
Stream pdfStream1 =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Sample2.pdf");
PdfLoadedDocument loadedDocument2 = new PdfLoadedDocument();
await loadedDocument2.OpenAsync(pdfStream1);
//Create the new document

```

```

PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
//Create a template from the first document
PdfPageBase loadedPage = loadedDocument1.Pages[0];
PdfTemplate template = loadedPage.CreateTemplate();
//Draw the loaded template into new document.
page.Graphics.DrawPdfTemplate(template, new PointF(0, 0), new SizeF(500,
700));
// Create a template from the second document
loadedPage = loadedDocument2.Pages[0];
template = loadedPage.CreateTemplate();
//Draw the loaded template into new document.
page.Graphics.DrawPdfTemplate(template, new PointF(10, 10), new SizeF(400,
500));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the documents
document.Close(true);
loadedDocument1.Close(true);
loadedDocument2.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream1 = new FileStream(fileName1, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument1 = new PdfLoadedDocument(docStream1);
//Load the PDF document
FileStream docStream2 = new FileStream(fileName2, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument2 = new PdfLoadedDocument(docStream2);
//Create the new document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
//Create a template from the first document
PdfPageBase loadedPage = loadedDocument1.Pages[0];
PdfTemplate template = loadedPage.CreateTemplate();
//Draw the loaded template into new document.
page.Graphics.DrawPdfTemplate(template, new PointF(0, 0), new SizeF(500,
700));
// Create a template from the second document
loadedPage = loadedDocument2.Pages[0];
template = loadedPage.CreateTemplate();
//Draw the loaded template into new document.
page.Graphics.DrawPdfTemplate(template, new PointF(10, 10), new SizeF(400,
500));
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the documents
document.Close(true);

```

```
loadedDocument1.Close(true);
loadedDocument2.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the existing document
Stream docStream1 =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample1.pdf");
PdfLoadedDocument loadedDocument1 = new PdfLoadedDocument(docStream1);
Stream docStream2 =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample2.pdf");
PdfLoadedDocument loadedDocument2 = new PdfLoadedDocument(docStream2);
//Create the new document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
//Create a template from the first document
PdfPageBase loadedPage = loadedDocument1.Pages[0];
PdfTemplate template = loadedPage.CreateTemplate();
//Draw the loaded template into new document.
page.Graphics.DrawPdfTemplate(template, new PointF(0, 0), new SizeF(500,
700));
// Create a template from the second document
loadedPage = loadedDocument2.Pages[0];
template = loadedPage.CreateTemplate();
//Draw the loaded template into new document.
page.Graphics.DrawPdfTemplate(template, new PointF(10, 10), new SizeF(400,
500));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the documents
document.Close(true);
loadedDocument1.Close(true);
loadedDocument2.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
"application/pdf", stream);
}
else
{
```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
"application/pdf", stream);
}
```

Working with Headers and Footers

Essential PDF supports to draw the header and footer in PDF document using [PdfPageTemplateElement](#) class. The header and footer can contain any types of element including dynamic fields.

Adding an automatic field in header and footer

Essential PDF supports to add page count, page numbers, date and time using dynamic fields such as PdfPageCountField, PdfPageNumberField, etc.

The below code snippet explains how to draw the page numbers in footer using automatic fields.

C#

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a header and draw the image.
RectangleF bounds = new RectangleF(0, 0,
pdfDocument.Pages[0].GetClientSize().Width, 50);
PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
PdfImage image = new PdfBitmap(@"Logo.png");
//Draw the image in the header.
header.Graphics.DrawImage(image, new PointF(0, 0), new SizeF(100, 50));
//Add the header at the top.
pdfDocument.Template.Top = header;
//Create a Page template that can be used as footer.
PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds);
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 7);
PdfBrush brush = new PdfSolidBrush(Color.Black);
//Create page number field.
PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
//Create page count field.
PdfPageCountField count = new PdfPageCountField(font, brush);
//Add the fields in composite fields.
PdfCompositeField compositeField = new PdfCompositeField(font, brush, "Page
{0} of {1}", pageNumber, count);
compositeField.Bounds = footer.Bounds;
//Draw the composite field in footer.
compositeField.Draw(footer.Graphics, new PointF(470, 40));
//Add the footer template at the bottom.
pdfDocument.Template.Bottom = footer;
//Save and close the document.
pdfDocument.Save("Output.pdf");
pdfDocument.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim pdfDocument As New PdfDocument()
'Add a page to the PDF document.
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
```

```

'Create a header and draw the image.
Dim bounds As New RectangleF(0, 0,
pdfDocument.Pages(0).GetClientSize().Width, 50)
Dim header As New PdfPageTemplateElement(bounds)
Dim image As PdfImage = New PdfBitmap("Logo.jpg")
'Draw the image in the Header.
header.Graphics.DrawImage(image, New PointF(0, 0), New SizeF(100, 50))
'Add the header at the top.
pdfDocument.Template.Top = header
'Create a page template that can be used as footer.
Dim footer As New PdfPageTemplateElement(bounds)
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 7)
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
'Create page number field.
Dim pageNumber As New PdfPageNumberField(font, brush)
'Create page count field.
Dim count As New PdfPageCountField(font, brush)
'Add the fields in composite fields.
Dim compositeField As New PdfCompositeField(font, brush, "Page {0} of {1}",
pageNumber, count)
compositeField.Bounds = footer.Bounds
'Draw the composite field in footer.
compositeField.Draw(footer.Graphics, New PointF(470, 40))
'Add the footer template at the bottom.
pdfDocument.Template.Bottom = footer
'Save and close the document.
pdfDocument.Save("Output.pdf")
pdfDocument.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a header and draw the image.
RectangleF bounds = new RectangleF(0, 0,
pdfDocument.Pages[0].GetClientSize().Width, 50);
PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
//Load the image as stream
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Logo.png");
PdfImage image = new PdfBitmap(imageStream);
//Draw the image in the header.
header.Graphics.DrawImage(image, new PointF(0, 0), new SizeF(100, 50));
//Add the header at the top.
pdfDocument.Template.Top = header;
//Create a Page template that can be used as footer.
PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds);
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 7);
PdfBrush brush = new PdfSolidBrush(Color.FromArgb(255, 0, 0, 0));
//Create page number field.
PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
//Create page count field.
PdfPageCountField count = new PdfPageCountField(font, brush);

```



```
//Add the fields in composite fields.
PdfCompositeField compositeField = new PdfCompositeField(font, brush, "Page
{0} of {1}", pageNumber, count);
compositeField.Bounds = footer.Bounds;
//Draw the composite field in footer.
compositeField.Draw(footer.Graphics, new PointF(470, 40));
//Add the footer template at the bottom.
pdfDocument.Template.Bottom = footer;
//Save the PDF document to stream
MemoryStream ms = new MemoryStream();
await pdfDocument.SaveAsync(ms);
//Close the document
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(ms, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a header and draw the image.
RectangleF bounds = new RectangleF(0, 0,
pdfDocument.Pages[0].GetClientSize().Width, 50);
PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
//Load the PDF document
FileStream imageStream = new FileStream("Logo.png", FileMode.Open,
FileAccess.Read);
PdfImage image = new PdfBitmap(imageStream);
//Draw the image in the header.
header.Graphics.DrawImage(image, new PointF(0, 0), new SizeF(100, 50));
//Add the header at the top.
pdfDocument.Template.Top = header;
//Create a Page template that can be used as footer.
PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds);
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 7);
PdfBrush brush = new PdfSolidBrush(Color.Black);
//Create page number field.
PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
//Create page count field.
PdfPageCountField count = new PdfPageCountField(font, brush);
//Add the fields in composite fields.
PdfCompositeField compositeField = new PdfCompositeField(font, brush, "Page
{0} of {1}", pageNumber, count);
compositeField.Bounds = footer.Bounds;
//Draw the composite field in footer.
compositeField.Draw(footer.Graphics, new PointF(470, 40));
//Add the footer template at the bottom.
pdfDocument.Template.Bottom = footer;
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Closes the document
```

```
pdfDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
//Create a header and draw the image.
RectangleF bounds = new RectangleF(0, 0,
pdfDocument.Pages[0].GetClientSize().Width, 50);
PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
//Load the image file as stream
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Logo.png");
PdfImage image = new PdfBitmap(imageStream);
//Draw the image in the header.
header.Graphics.DrawImage(image, new PointF(0, 0), new SizeF(100, 50));
//Add the header at the top.
pdfDocument.Template.Top = header;
//Create a Page template that can be used as footer.
PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds);
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 7);
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Black);
//Create page number field.
PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
//Create page count field.
PdfPageCountField count = new PdfPageCountField(font, brush);
//Add the fields in composite fields.
PdfCompositeField compositeField = new PdfCompositeField(font, brush, "Page
{0} of {1}", pageNumber, count);
compositeField.Bounds = footer.Bounds;
//Draw the composite field in footer.
compositeField.Draw(footer.Graphics, new PointF(470, 40));
//Add the footer template at the bottom.
pdfDocument.Template.Bottom = footer;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Closes the document
pdfDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}
```

Working with Shapes

Essential PDF has support for adding the below shapes.

- Line
- Curve
- Path
- Text
- Rectangle
- Pie
- Arc
- Bezier
- Ellipse

Adding Shapes to a PDF document

Essential PDF supports adding shapes with different types of brushes like solid brush, gradient brush, tiling brush, and image brush along with various [color spaces](#) and transparency levels.

Polygon

You can draw a polygon in PDF document by using the [DrawPolygon](#) method of [PdfGraphics](#). The following code snippet explains how to draw a polygon in new PDF document.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfPen to draw the polygon
PdfPen pen = new PdfPen(PdfBrushes.Brown, 10f);
//Initialize PdfLinearGradientBrush for drawing the polygon
PdfLinearGradientBrush brush = new PdfLinearGradientBrush(new PointF(10,
100), new PointF(100, 200), new PdfColor(Color.Red), new
PdfColor(Color.Green));
//Create the polygon points
PointF p1 = new PointF(10, 100);
PointF p2 = new PointF(10, 200);
PointF p3 = new PointF(100, 100);
PointF p4 = new PointF(100, 200);
PointF p5 = new PointF(55, 150);
PointF[] points = { p1, p2, p3, p4, p5 };
//Draw the polygon on PDF document
page.Graphics.DrawPolygon(pen, brush, points);
//Save the PDF document
document.Save("Output.pdf");
```

```
//Close the instance of PdfDocument
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = document.Pages.Add
'Initialize PdfPen to draw the polygon
Dim pen As PdfPen = New PdfPen(PdfBrushes.Brown, 10.0F)
'Initialize PdfLinearGradientBrush for drawing the polygon
Dim brush As PdfLinearGradientBrush = New PdfLinearGradientBrush(New
PointF(10, 100), New PointF(100, 200), New PdfColor(Color.Red), New
PdfColor(Color.Green))
'Create the polygon points
Dim p1 As PointF = New PointF(10, 100)
Dim p2 As PointF = New PointF(10, 200)
Dim p3 As PointF = New PointF(100, 100)
Dim p4 As PointF = New PointF(100, 200)
Dim p5 As PointF = New PointF(55, 150)
Dim points As PointF() = {p1, p2, p3, p4, p5}
'Draw the polygon on PDF document
page.Graphics.DrawPolygon(pen, brush, points)
'Save the PDF document
document.Save("Output.pdf")
'Close the instance of PdfDocument
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfPen to draw the polygon
PdfPen pen = new PdfPen(PdfBrushes.Brown, 10f);
//Initialize PdfLinearGradientBrush for drawing the polygon
PdfLinearGradientBrush brush = new PdfLinearGradientBrush(new PointF(10,
100), new PointF(100, 200), new PdfColor(Color.FromArgb(255, 255, 0, 0)),
new PdfColor(Color.FromArgb(255, 0, 128, 0)));
//Create the polygon points
PointF p1 = new PointF(10, 100);
PointF p2 = new PointF(10, 200);
PointF p3 = new PointF(100, 100);
PointF p4 = new PointF(100, 200);
PointF p5 = new PointF(55, 150);
PointF[] points = { p1, p2, p3, p4, p5 };
//Draw the polygon on PDF document
page.Graphics.DrawPolygon(pen, brush, points);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
```

```
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfPen to draw the polygon
PdfPen pen = new PdfPen(PdfBrushes.Brown, 10f);
//Initialize PdfLinearGradientBrush for drawing the polygon
PdfLinearGradientBrush brush = new PdfLinearGradientBrush(new PointF(10,
100), new PointF(100, 200), new PdfColor(Color.Red), new
PdfColor(Color.Green));
//Create the polygon points
PointF p1 = new PointF(10, 100);
PointF p2 = new PointF(10, 200);
PointF p3 = new PointF(100, 100);
PointF p4 = new PointF(100, 200);
PointF p5 = new PointF(55, 150);
PointF[] points = { p1, p2, p3, p4, p5 };
//Draw the polygon on PDF document
page.Graphics.DrawPolygon(pen, brush, points);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfPen to draw the polygon
PdfPen pen = new PdfPen(PdfBrushes.Brown, 10f);
//Initialize PdfLinearGradientBrush for drawing the polygon
PdfLinearGradientBrush brush = new PdfLinearGradientBrush(new PointF(10,
100), new PointF(100, 200), new
PdfColor(Syncfusion.Drawing.Color.FromArgb(255, 255, 0, 0)), new
PdfColor(Syncfusion.Drawing.Color.FromArgb(255, 0, 128, 0)));
//Create the polygon points
PointF p1 = new PointF(10, 100);
PointF p2 = new PointF(10, 200);
PointF p3 = new PointF(100, 100);
PointF p4 = new PointF(100, 200);
PointF p5 = new PointF(55, 150);
PointF[] points = { p1, p2, p3, p4, p5 };
```

```

//Draw the polygon on PDF document
page.Graphics.DrawPolygon(pen, brush, points);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following code snippet explains how to draw a polygon in an existing PDF document.

C#

```

//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize PdfPen to draw the polygon
PdfPen pen = new PdfPen(PdfBrushes.Brown, 10f);
//Initialize PdfLinearGradientBrush for drawing the polygon
PdfLinearGradientBrush brush = new PdfLinearGradientBrush(new PointF(10,
100), new PointF(100, 200), new PdfColor(Color.Red), new
PdfColor(Color.Green));
//Create the polygon points
PointF p1 = new PointF(10, 100);
PointF p2 = new PointF(10, 200);
PointF p3 = new PointF(100, 100);
PointF p4 = new PointF(100, 200);
PointF p5 = new PointF(55, 150);
PointF[] points = { p1, p2, p3, p4, p5 };
//Draw the polygon on PDF document
loadedPage.Graphics.DrawPolygon(pen, brush, points);
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);

```

VB.NET

```

'Load an existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the page into PdfLoadedPage

```

```

Dim loadedPage As PdfLoadedPage = CType(loadedDocument.Pages(0),
PdfLoadedPage)
'Initialize PdfPen to draw the polygon
Dim pen As PdfPen = New PdfPen(PdfBrushes.Brown, 10.0F)
'Initialize PdfLinearGradientBrush for drawing the polygon
Dim brush As PdfLinearGradientBrush = New PdfLinearGradientBrush(New
PointF(10, 100), New PointF(100, 200), New PdfColor(Color.Red), New
PdfColor(Color.Green))
'Create the polygon points
Dim p1 As PointF = New PointF(10, 100)
Dim p2 As PointF = New PointF(10, 200)
Dim p3 As PointF = New PointF(100, 100)
Dim p4 As PointF = New PointF(100, 200)
Dim p5 As PointF = New PointF(55, 150)
Dim points As PointF() = {p1, p2, p3, p4, p5}
'Draw the polygon on PDF document
loadedPage.Graphics.DrawPolygon(pen, brush, points)
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the instance of PdfLoadedDocument
loadedDocument.Close(True)

```

UWP

```

//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize PdfPen to draw the polygon
PdfPen pen = new PdfPen(PdfBrushes.Brown, 10f);
//Initialize PdfLinearGradientBrush for drawing the polygon
PdfLinearGradientBrush brush = new PdfLinearGradientBrush(new PointF(10,
100), new PointF(100, 200), new PdfColor(Color.FromArgb(255, 255, 0, 0)),
new PdfColor(Color.FromArgb(255, 0, 128, 0)));
//Create the polygon points
PointF p1 = new PointF(10, 100);
PointF p2 = new PointF(10, 200);
PointF p3 = new PointF(100, 100);
PointF p4 = new PointF(100, 200);
PointF p5 = new PointF(55, 150);
PointF[] points = { p1, p2, p3, p4, p5 };
//Draw the polygon on PDF document
loadedPage.Graphics.DrawPolygon(pen, brush, points);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```
//Load the PDF document as stream
FileStream inputStream = new FileStream("Input.pdf", FileMode.Open);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize PdfPen to draw the polygon
PdfPen pen = new PdfPen(PdfBrushes.Brown, 10f);
//Initialize PdfLinearGradientBrush for drawing the polygon
PdfLinearGradientBrush brush = new PdfLinearGradientBrush(new PointF(10,
100), new PointF(100, 200), new PdfColor(Color.Red), new
PdfColor(Color.Green));
//Create the polygon points
PointF p1 = new PointF(10, 100);
PointF p2 = new PointF(10, 200);
PointF p3 = new PointF(100, 100);
PointF p4 = new PointF(100, 200);
PointF p5 = new PointF(55, 150);
PointF[] points = { p1, p2, p3, p4, p5 };
//Draw the polygon on PDF document
loadedPage.Graphics.DrawPolygon(pen, brush, points);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize PdfPen to draw the polygon
PdfPen pen = new PdfPen(PdfBrushes.Brown, 10f);
//Initialize PdfLinearGradientBrush for drawing the polygon
PdfLinearGradientBrush brush = new PdfLinearGradientBrush(new PointF(10,
100), new PointF(100, 200), new
PdfColor(Syncfusion.Drawing.Color.FromArgb(255, 255, 0, 0)), new
PdfColor(Syncfusion.Drawing.Color.FromArgb(255, 0, 128, 0)));
//Create the polygon points
PointF p1 = new PointF(10, 100);
PointF p2 = new PointF(10, 200);
PointF p3 = new PointF(100, 100);
PointF p4 = new PointF(100, 200);
PointF p5 = new PointF(55, 150);
PointF[] points = { p1, p2, p3, p4, p5 };
```



```

//Draw the polygon on PDF document
loadedPage.Graphics.DrawPolygon(pen, brush, points);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Line

You can draw a line in PDF document by using the [DrawLine](#) method of [PdfGraphics](#). The following code snippet explains how to draw a line in new PDF document.

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize pen to draw the line
PdfPen pen = new PdfPen(PdfBrushes.Black, 5f);
//Create the line points
PointF point1 = new PointF(10, 10);
PointF point2 = new PointF(10, 100);
//Draw the line on PDF document
page.Graphics.DrawLine(pen, point1, point2);
//Save the PDF document
document.Save("Output.pdf");
//Close the instance of PdfDocument
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = document.Pages.Add
'Initialize pen to draw the line
Dim pen As PdfPen = New PdfPen(PdfBrushes.Black, 5.0F)
'Create the line points
Dim point1 As PointF = New PointF(10, 10)

```

```

Dim point2 As PointF = New PointF(10, 100)
'Draw the line on PDF document
page.Graphics.DrawLine(pen, point1, point2)
'Save the PDF document
document.Save("Output.pdf")
'Close the instance of PdfDocument
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize pen to draw the line
PdfPen pen = new PdfPen(PdfBrushes.Black, 5f);
//Create the line points
PointF point1 = new PointF(10, 10);
PointF point2 = new PointF(10, 100);
//Draw the line on PDF document
page.Graphics.DrawLine(pen, point1, point2);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize pen to draw the line
PdfPen pen = new PdfPen(PdfBrushes.Black, 5f);
//Create the line points
PointF point1 = new PointF(10, 10);
PointF point2 = new PointF(10, 100);
//Draw the line on PDF document
page.Graphics.DrawLine(pen, point1, point2);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileName = "Output.pdf";
return fileStreamResult;

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize pen to draw the line
PdfPen pen = new PdfPen(PdfBrushes.Black, 5f);
//Create the line points
PointF point1 = new PointF(10, 10);
PointF point2 = new PointF(10, 100);
//Draw the line on PDF document
page.Graphics.DrawLine(pen, point1, point2);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following code snippet explains how to draw a line in an existing PDF document.

C#

```

//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize pen to draw the line
PdfPen pen = new PdfPen(PdfBrushes.Black, 5f);
//Create the line points
PointF point1 = new PointF(10, 10);
PointF point2 = new PointF(10, 100);
//Draw the line on PDF document
loadedPage.Graphics.DrawLine(pen, point1, point2);
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);

```

VB.NET

```

'Load an existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the page into PdfLoadedPage
Dim loadedPage As PdfLoadedPage = CType(loader.Document.Pages(0), PdfLoadedPage)
'Initialize pen to draw the line
Dim pen As PdfPen = New PdfPen(PdfBrushes.Black, 5.0F)
'Create the line points
Dim point1 As PointF = New PointF(10, 10)
Dim point2 As PointF = New PointF(10, 100)
'Draw the line on PDF document
loadedPage.Graphics.DrawLine(pen, point1, point2)
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the instance of PdfLoadedDocument
loadedDocument.Close(True)

```

UWP

```

//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize pen to draw the line
PdfPen pen = new PdfPen(PdfBrushes.Black, 5f);
//Create the line points
PointF point1 = new PointF(10, 10);
PointF point2 = new PointF(10, 100);
//Draw the line on PDF document
loadedPage.Graphics.DrawLine(pen, point1, point2);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document as stream
FileStream inputStream = new FileStream("Input.pdf", FileMode.Open);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize pen to draw the line
PdfPen pen = new PdfPen(PdfBrushes.Black, 5f);
//Create the line points
PointF point1 = new PointF(10, 10);
PointF point2 = new PointF(10, 100);
//Draw the line on PDF document

```

```
loadedPage.Graphics.DrawLine(pen, point1, point2);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize pen to draw the line
PdfPen pen = new PdfPen(PdfBrushes.Black, 5f);
//Create the line points
PointF point1 = new PointF(10, 10);
PointF point2 = new PointF(10, 100);
//Draw the line on PDF document
loadedPage.Graphics.DrawLine(pen, point1, point2);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Curve

You can draw a curve in PDF document by using the [Draw](#) method of [PdfBezierCurve](#). The following code snippet explains how to draw a curve in new PDF document.

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfGraphics for PdfPage
PdfGraphics graphics = page.Graphics;
//Create new instance of PdfBezierCurve
PdfBezierCurve bezier = new PdfBezierCurve(new PointF(0, 0), new PointF(100,
50), new PointF(50, 50), new PointF(100, 100));
//Draw the bezier curve on PDF document
bezier.Draw(graphics, new PointF(10, 10));
//Save the PDF document
document.Save("Output.pdf");
//Close the instance of PdfDocument
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = document.Pages.Add
'Initialize PdfGraphics for PdfPage
Dim graphics As PdfGraphics = page.Graphics
'Create new instance of PdfBezierCurve
Dim bezier As PdfBezierCurve = New PdfBezierCurve(New PointF(0, 0), New
PointF(100, 50), New PointF(50, 50), New PointF(100, 100))
'Draw the bezier curve on PDF document
bezier.Draw(graphics, New PointF(10, 10))
'Save the PDF document
document.Save("Output.pdf")
'Close the instance of PdfDocument
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfGraphics for PdfPage
PdfGraphics graphics = page.Graphics;
//Create new instance of PdfBezierCurve
PdfBezierCurve bezier = new PdfBezierCurve(new PointF(0, 0), new PointF(100,
50), new PointF(50, 50), new PointF(100, 100));
//Draw the bezier curve on PDF document
bezier.Draw(graphics, new PointF(10, 10));
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfGraphics for PdfPage
PdfGraphics graphics = page.Graphics;
//Create new instance of PdfBezierCurve
PdfBezierCurve bezier = new PdfBezierCurve(new PointF(0, 0), new PointF(100,
50), new PointF(50, 50), new PointF(100, 100));
//Draw the bezier curve on PDF document
bezier.Draw(graphics, new PointF(10, 10));
//Save the PDF document to MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfGraphics for PdfPage
PdfGraphics graphics = page.Graphics;
//Create new instance of PdfBezierCurve
PdfBezierCurve bezier = new PdfBezierCurve(new PointF(0, 0), new PointF(100,
50), new PointF(50, 50), new PointF(100, 100));
//Draw the bezier curve on PDF document
bezier.Draw(graphics, new PointF(10, 10));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

The following code snippet explains how to draw a curve in an existing PDF document.

C#

```
//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the graphics of PdfLoadedPage
PdfGraphics graphics = loadedPage.Graphics;
//Create new instance of PdfBezierCurve
PdfBezierCurve bezier = new PdfBezierCurve(new PointF(0, 0), new PointF(100,
50), new PointF(50, 50), new PointF(100, 100));
//Draw the bezier curve on PDF document
bezier.Draw(graphics, new PointF(10, 10));
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
```

VB.NET

```
'Load an existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the page into PdfLoadedPage
Dim loadedPage As PdfLoadedPage = CType(loadedDocument.Pages(0),
PdfLoadedPage)
'Get the graphics of PdfLoadedPage
Dim graphics As PdfGraphics = loadedPage.Graphics
'Create new instance of PdfBezierCurve
Dim bezier As PdfBezierCurve = New PdfBezierCurve(New PointF(0, 0), New
PointF(100, 50), New PointF(50, 50), New PointF(100, 100))
'Draw the bezier curve on PDF document
bezier.Draw(graphics, New PointF(10, 10))
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the instance of PdfLoadedDocument
loadedDocument.Close(True)
```

UWP

```
//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the graphics of PdfLoadedPage
PdfGraphics graphics = loadedPage.Graphics;
//Create new instance of PdfBezierCurve
```



```

PdfBezierCurve bezier = new PdfBezierCurve(new PointF(0, 0), new PointF(100,
50), new PointF(50, 50), new PointF(100, 100));
//Draw the bezier curve on PDF document
bezier.Draw(graphics, new PointF(10, 10));
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document as stream
FileStream inputStream = new FileStream("Input.pdf", FileMode.Open);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the graphics of PdfLoadedPage
PdfGraphics graphics = loadedPage.Graphics;
//Create new instance of PdfBezierCurve
PdfBezierCurve bezier = new PdfBezierCurve(new PointF(0, 0), new PointF(100,
50), new PointF(50, 50), new PointF(100, 100));
//Draw the bezier curve on PDF document
bezier.Draw(graphics, new PointF(10, 10));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownloadName = "Output.pdf";
return fileStreamResult;

```

XAMARIN

```

//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the graphics of PdfLoadedPage
PdfGraphics graphics = loadedPage.Graphics;
//Create new instance of PdfBezierCurve
PdfBezierCurve bezier = new PdfBezierCurve(new PointF(0, 0), new PointF(100,
50), new PointF(50, 50), new PointF(100, 100));
//Draw the bezier curve on PDF document
bezier.Draw(graphics, new PointF(10, 10));
//Save the document to the stream

```

```

MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Path

You can draw a path in PDF document by using the [DrawPath](#) method of [PdfGraphics](#). The following code snippet explains how to draw path in a new PDF document.

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize a new PDF path
PdfPath path = new PdfPath();
//Add line path points
path.AddLine(new PointF(10, 100), new PointF(10, 200));
path.AddLine(new PointF(10, 200), new PointF(100, 100));
path.AddLine(new PointF(100, 100), new PointF(100, 200));
path.AddLine(new PointF(100, 200), new PointF(10, 100));
//Draw the PDF path on page
page.Graphics.DrawPath(PdfPens.Black, path);
//Save the PDF document
document.Save("Output.pdf");
//Close the instance of PdfDocument
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = document.Pages.Add
'Initialize a new PDF path
Dim path As New PdfPath()
'Add line path points
path.AddLine(New PointF(10, 100), New PointF(10, 200))
path.AddLine(New PointF(10, 200), New PointF(100, 100))

```

```

path.AddLine(New PointF(100, 100), New PointF(100, 200))
path.AddLine(New PointF(100, 200), New PointF(10, 100))
'Draw the PDF path on page
page.Graphics.DrawPath(PdfPens.Black, path)
'Save the PDF document
document.Save("Output.pdf")
'Close the instance of PdfDocument
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize a new PDF path
PdfPath path = new PdfPath();
//Add line path points
path.AddLine(new PointF(10, 100), new PointF(10, 200));
path.AddLine(new PointF(10, 200), new PointF(100, 100));
path.AddLine(new PointF(100, 100), new PointF(100, 200));
path.AddLine(new PointF(100, 200), new PointF(10, 100));
//Draw the PDF path on page
page.Graphics.DrawPath(PdfPens.Black, path);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize a new PDF path
PdfPath path = new PdfPath();
//Add line path points
path.AddLine(new PointF(10, 100), new PointF(10, 200));
path.AddLine(new PointF(10, 200), new PointF(100, 100));
path.AddLine(new PointF(100, 100), new PointF(100, 200));
path.AddLine(new PointF(100, 200), new PointF(10, 100));
//Draw the PDF path on page
page.Graphics.DrawPath(PdfPens.Black, path);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser

```

```

FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileName = "Output.pdf";
return fileStreamResult;

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize a new PDF path
PdfPath path = new PdfPath();
//Add line path points
path.AddLine(new PointF(10, 100), new PointF(10, 200));
path.AddLine(new PointF(10, 200), new PointF(100, 100));
path.AddLine(new PointF(100, 100), new PointF(100, 200));
path.AddLine(new PointF(100, 200), new PointF(10, 100));
//Draw the PDF path on page
page.Graphics.DrawPath(PdfPens.Black, path);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following code snippet explains how to draw path in an existing PDF document.

C#

```

//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize a new PDF path
PdfPath path = new PdfPath();
//Add line path points
path.AddLine(new PointF(10, 100), new PointF(10, 200));
path.AddLine(new PointF(10, 200), new PointF(100, 100));
path.AddLine(new PointF(100, 100), new PointF(100, 200));
path.AddLine(new PointF(100, 200), new PointF(10, 100));

```

```
//Draw the PDF path on page
loadedPage.Graphics.DrawPath(PdfPens.Black, path);
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
```

VB.NET

```
'Load an existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the page into PdfLoadedPage
Dim loadedPage As PdfLoadedPage = CType(loadedDocument.Pages(0), PdfLoadedPage)
'Initialize a new PDF path
Dim path As New PdfPath()
'Add line path points
path.AddLine(New PointF(10, 100), New PointF(10, 200))
path.AddLine(New PointF(10, 200), New PointF(100, 100))
path.AddLine(New PointF(100, 100), New PointF(100, 200))
path.AddLine(New PointF(100, 200), New PointF(10, 100))
'Draw the PDF path on page
loadedPage.Graphics.DrawPath(PdfPens.Black, path)
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the instance of PdfLoadedDocument
loadedDocument.Close(True)
```

UWP

```
//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize a new PDF path
PdfPath path = new PdfPath();
//Add line path points
path.AddLine(new PointF(10, 100), new PointF(10, 200));
path.AddLine(new PointF(10, 200), new PointF(100, 100));
path.AddLine(new PointF(100, 100), new PointF(100, 200));
path.AddLine(new PointF(100, 200), new PointF(10, 100));
//Draw the PDF path on page
loadedPage.Graphics.DrawPath(PdfPens.Black, path);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document as stream
FileStream inputStream = new FileStream("Input.pdf", FileMode.Open);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize a new PDF path
PdfPath path = new PdfPath();
//Add line path points
path.AddLine(new PointF(10, 100), new PointF(10, 200));
path.AddLine(new PointF(10, 200), new PointF(100, 100));
path.AddLine(new PointF(100, 100), new PointF(100, 200));
path.AddLine(new PointF(100, 200), new PointF(10, 100));
//Draw the PDF path on page
loadedPage.Graphics.DrawPath(PdfPens.Black, path);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize a new PDF path
PdfPath path = new PdfPath();
//Add line path points
path.AddLine(new PointF(10, 100), new PointF(10, 200));
path.AddLine(new PointF(10, 200), new PointF(100, 100));
path.AddLine(new PointF(100, 100), new PointF(100, 200));
path.AddLine(new PointF(100, 200), new PointF(10, 100));
//Draw the PDF path on page
loadedPage.Graphics.DrawPath(PdfPens.Black, path);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Text

You can draw text in a PDF document by using the [DrawString](#) method of [PdfGraphics](#). To draw different PDF elements based on the position of previously drawn element, draw the text by using [Draw](#) method of [PdfTextElement](#) and store its bounds in [PdfLayoutResult](#). Refer to the [Working with Text](#) section for more information.

Rectangle

You can draw a rectangle in PDF document by using the [DrawRectangle](#) method of [PdfGraphics](#). The following code snippet explains how to draw a rectangle in new PDF document.

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfSolidBrush for drawing the rectangle
PdfSolidBrush brush = new PdfSolidBrush(Color.Green);
//Set the bounds for rectangle
RectangleF bounds = new RectangleF(10, 10, 100, 50);
//Draw the rectangle on PDF document
page.Graphics.DrawRectangle(brush, bounds);
//Save the PDF document
document.Save("Output.pdf");
//Close the instance of PdfDocument
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = document.Pages.Add
'Initialize PdfSolidBrush for drawing the rectangle
Dim brush As PdfSolidBrush = New PdfSolidBrush(Color.Green)
'Set the bounds for rectangle
Dim bounds As RectangleF = New RectangleF(10, 10, 100, 50)
'Draw the rectangle on PDF document
page.Graphics.DrawRectangle(brush, bounds)
'Save the PDF document
document.Save("Output.pdf")
'Close the instance of PdfDocument
document.Close(True)

```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfSolidBrush for drawing the rectangle
PdfSolidBrush brush = new PdfSolidBrush(Color.FromArgb(255, 0, 128, 0));
//Set the bounds for rectangle
RectangleF bounds = new RectangleF(10, 10, 100, 50);
//Draw the rectangle on PDF document
page.Graphics.DrawRectangle(brush, bounds);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfSolidBrush for drawing the rectangle
PdfSolidBrush brush = new PdfSolidBrush(Color.Green);
//Set the bounds for rectangle
RectangleF bounds = new RectangleF(10, 10, 100, 50);
//Draw the rectangle on PDF document
page.Graphics.DrawRectangle(brush, bounds);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfSolidBrush for drawing the rectangle
PdfSolidBrush brush = new
PdfSolidBrush(Syncfusion.Drawing.Color.FromArgb(255, 0, 128, 0));
```



```

//Set the bounds for rectangle
RectangleF bounds = new RectangleF(10, 10, 100, 50);
//Draw the rectangle on PDF document
page.Graphics.DrawRectangle(brush, bounds);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following code snippet explains how to draw a rectangle in an existing PDF document.

C#

```

//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize PdfSolidBrush for drawing the rectangle
PdfSolidBrush brush = new PdfSolidBrush(Color.Green);
//Set the bounds for rectangle
RectangleF bounds = new RectangleF(10, 10, 100, 50);
//Draw the rectangle on PDF document
loadedPage.Graphics.DrawRectangle(brush, bounds);
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);

```

VB.NET

```

'Load an existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the page into PdfLoadedPage
Dim loadedPage As PdfLoadedPage = CType(loadedDocument.Pages(0),
PdfLoadedPage)
'Initialize PdfSolidBrush for drawing the rectangle
Dim brush As PdfSolidBrush = New PdfSolidBrush(Color.Green)
'Set the bounds for rectangle
Dim bounds As RectangleF = New RectangleF(10, 10, 100, 50)
'Draw the rectangle on PDF document

```

```
loadedPage.Graphics.DrawRectangle(brush, bounds)
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the instance of PdfLoadedDocument
loadedDocument.Close(True)
```

UWP

```
//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize PdfSolidBrush for drawing the rectangle
PdfSolidBrush brush = new PdfSolidBrush(Color.FromArgb(255, 0, 128, 0));
//Set the bounds for rectangle
RectangleF bounds = new RectangleF(10, 10, 100, 50);
//Draw the rectangle on PDF document
loadedPage.Graphics.DrawRectangle(brush, bounds);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document as stream
FileStream inputStream = new FileStream("Input.pdf", FileMode.Open);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize PdfSolidBrush for drawing the rectangle
PdfSolidBrush brush = new PdfSolidBrush(Color.Green);
//Set the bounds for rectangle
RectangleF bounds = new RectangleF(10, 10, 100, 50);
//Draw the rectangle on PDF document
loadedPage.Graphics.DrawRectangle(brush, bounds);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize PdfSolidBrush for drawing the rectangle
PdfSolidBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Green);
//Set the bounds for rectangle
RectangleF bounds = new RectangleF(10, 10, 100, 50);
//Draw the rectangle on PDF document
loadedPage.Graphics.DrawRectangle(brush, bounds);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Pie

You can draw a pie in PDF document by using the [DrawPie](#) method of [PdfGraphics](#). The following code snippet explains how to draw a pie in new PDF document.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize the pen for drawing pie
PdfPen pen = new PdfPen(PdfBrushes.Brown, 5f);
//Set the line join style of the pen
pen.LineJoin = PdfLineJoin.Round;
//Set the bounds for pie
RectangleF rectangle = new RectangleF(10, 50, 200, 200);
//Draw the pie on PDF document
page.Graphics.DrawPie(pen, PdfBrushes.Green, rectangle, 180, 60);
//Save the PDF document
document.Save("Output.pdf");
```

```
//Close the instance of PdfDocument
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = document.Pages.Add
'Initialize the pen for drawing pie
Dim pen As PdfPen = New PdfPen(PdfBrushes.Brown, 5.0F)
'Set the line join style of the pen
pen.LineJoin = PdfLineJoin.Round
'Set the bounds for pie
Dim rectangle As RectangleF = New RectangleF(10, 50, 200, 200)
'Draw the pie on PDF document
page.Graphics.DrawPie(pen, PdfBrushes.Green, rectangle, 180, 60)
'Save the PDF document
document.Save("Output.pdf")
'Close the instance of PdfDocument
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize the pen for drawing pie
PdfPen pen = new PdfPen(PdfBrushes.Brown, 5f);
//Set the line join style of the pen
pen.LineJoin = PdfLineJoin.Round;
//Set the bounds for pie
RectangleF rectangle = new RectangleF(10, 50, 200, 200);
//Draw the pie on PDF document
page.Graphics.DrawPie(pen, PdfBrushes.Green, rectangle, 180, 60);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize the pen for drawing pie
PdfPen pen = new PdfPen(PdfBrushes.Brown, 5f);
//Set the line join style of the pen
```

```

pen.LineJoin = PdfLineJoin.Round;
//Set the bounds for pie
RectangleF rectangle = new RectangleF(10, 50, 200, 200);
//Draw the pie on PDF document
page.Graphics.DrawPie(pen, PdfBrushes.Green, rectangle, 180, 60);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize the pen for drawing pie
PdfPen pen = new PdfPen(PdfBrushes.Brown, 5f);
//Set the line join style of the pen
pen.LineJoin = PdfLineJoin.Round;
//Set the bounds for pie
RectangleF rectangle = new RectangleF(10, 50, 200, 200);
//Draw the pie on PDF document
page.Graphics.DrawPie(pen, PdfBrushes.Green, rectangle, 180, 60);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following code snippet explains how to draw a pie in an existing PDF document.

C#

```

//Load an existing PDF document

```

```

PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize the pen for drawing pie
PdfPen pen = new PdfPen(PdfBrushes.Brown, 5f);
//Set the line join style of the pen
pen.LineJoin = PdfLineJoin.Round;
//Set the bounds for pie
RectangleF rectangle = new RectangleF(10, 50, 200, 200);
//Draw the pie on PDF document
loadedPage.Graphics.DrawPie(pen, PdfBrushes.Green, rectangle, 180, 60);
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);

```

VB.NET

```

'Load an existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the page into PdfLoadedPage
Dim loadedPage As PdfLoadedPage = CType(loadedDocument.Pages(0), PdfLoadedPage)
'Initialize the pen for drawing pie
Dim pen As PdfPen = New PdfPen(PdfBrushes.Brown, 5.0F)
'Set the line join style of the pen
pen.LineJoin = PdfLineJoin.Round
'Set the bounds for pie
Dim rectangle As RectangleF = New RectangleF(10, 50, 200, 200)
'Draw the pie on PDF document
loadedPage.Graphics.DrawPie(pen, PdfBrushes.Green, rectangle, 180, 60)
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the instance of PdfLoadedDocument
loadedDocument.Close(True)

```

UWP

```

//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize the pen for drawing pie
PdfPen pen = new PdfPen(PdfBrushes.Brown, 5f);
//Set the line join style of the pen
pen.LineJoin = PdfLineJoin.Round;
//Set the bounds for pie
RectangleF rectangle = new RectangleF(10, 50, 200, 200);
//Draw the pie on PDF document
loadedPage.Graphics.DrawPie(pen, PdfBrushes.Green, rectangle, 180, 60);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it

```

```
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document as stream
FileStream inputStream = new FileStream("Input.pdf", FileMode.Open);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize the pen for drawing pie
PdfPen pen = new PdfPen(PdfBrushes.Brown, 5f);
//Set the line join style of the pen
pen.LineJoin = PdfLineJoin.Round;
//Set the bounds for pie
RectangleF rectangle = new RectangleF(10, 50, 200, 200);
//Draw the pie on PDF document
loadedPage.Graphics.DrawPie(pen, PdfBrushes.Green, rectangle, 180, 60);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileName = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize the pen for drawing pie
PdfPen pen = new PdfPen(PdfBrushes.Brown, 5f);
//Set the line join style of the pen
pen.LineJoin = PdfLineJoin.Round;
//Set the bounds for pie
RectangleF rectangle = new RectangleF(10, 50, 200, 200);
//Draw the pie on PDF document
loadedPage.Graphics.DrawPie(pen, PdfBrushes.Green, rectangle, 180, 60);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream into PDF file
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Arc

You can draw an arc in PDF document by using the [DrawArc](#) method of [PdfGraphics](#). The following code snippet explains how to draw an arc in new PDF document.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize the pen for drawing an arc
PdfPen pen = new PdfPen(Color.Brown, 10f);
//Set the line join style of the pen
pen.LineCap = PdfLineCap.Square;
//Set the bounds for arc
RectangleF bounds = new RectangleF(20, 40, 200, 200);
//Draw the arc on PDF document
page.Graphics.DrawArc(pen, bounds, 270, 90);
//Save the PDF document
document.Save("Output.pdf");
//Close the instance of PdfDocument
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = document.Pages.Add
'Initialize the pen for drawing an arc
Dim pen As PdfPen = New PdfPen(Color.Brown, 10.0F)
'Set the line join style of the pen
pen.LineCap = PdfLineCap.Square
'Set the bounds for arc
Dim bounds As RectangleF = New RectangleF(20, 40, 200, 200)
'Draw the arc on PDF document
page.Graphics.DrawArc(pen, bounds, 270, 90)
'Save the PDF document
document.Save("Output.pdf")
'Close the instance of PdfDocument
```



```
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize the pen for drawing an arc
PdfPen pen = new PdfPen(Color.FromArgb(255, 165, 42, 42), 10f);
//Set the line join style of the pen
pen.LineCap = PdfLineCap.Square;
//Set the bounds for arc
RectangleF bounds = new RectangleF(20, 40, 200, 200);
//Draw the arc on PDF document
page.Graphics.DrawArc(pen, bounds, 270, 90);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize the pen for drawing an arc
PdfPen pen = new PdfPen(Color.Brown, 10f);
//Set the line join style of the pen
pen.LineCap = PdfLineCap.Square;
//Set the bounds for arc
RectangleF bounds = new RectangleF(20, 40, 200, 200);
//Draw the arc on PDF document
page.Graphics.DrawArc(pen, bounds, 270, 90);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
```

```

//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize the pen for drawing an arc
PdfPen pen = new PdfPen(Syncfusion.Drawing.Color.Brown, 10f);
//Set the line join style of the pen
pen.LineCap = PdfLineCap.Square;
//Set the bounds for arc
RectangleF bounds = new RectangleF(20, 40, 200, 200);
//Draw the arc on PDF document
page.Graphics.DrawArc(pen, bounds, 270, 90);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following code snippet explains how to draw an arc in an existing PDF document.

C#

```

//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize the pen for drawing an arc
PdfPen pen = new PdfPen(Color.Brown, 10f);
//Set the line join style of the pen
pen.LineCap = PdfLineCap.Square;
//Set the bounds for arc
RectangleF bounds = new RectangleF(20, 40, 200, 200);
//Draw the arc on PDF document
loadedPage.Graphics.DrawArc(pen, bounds, 270, 90);
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);

```

VB.NET

```

'Load an existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")

```

```

'Get the page into PdfLoadedPage
Dim loadedPage As PdfLoadedPage = CType(loader.Document.Pages(0),
PdfLoadedPage)
'Initialize the pen for drawing an arc
Dim pen As PdfPen = New PdfPen(Color.Brown, 10.0F)
'Set the line join style of the pen
pen.LineCap = PdfLineCap.Square
'Set the bounds for arc
Dim bounds As RectangleF = New RectangleF(20, 40, 200, 200)
'Draw the arc on PDF document
loadedPage.Graphics.DrawArc(pen, bounds, 270, 90)
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the instance of PdfLoadedDocument
loadedDocument.Close(True)

```

UWP

```

//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream(Sample.Ass
ets.Input.pdf);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize the pen for drawing an arc
PdfPen pen = new PdfPen(Color.FromArgb(255, 165, 42, 42), 10f);
//Set the line join style of the pen
pen.LineCap = PdfLineCap.Square;
//Set the bounds for arc
RectangleF bounds = new RectangleF(20, 40, 200, 200);
//Draw the arc on PDF document
loadedPage.Graphics.DrawArc(pen, bounds, 270, 90);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document as stream
FileStream inputStream = new FileStream("Input.pdf", FileMode.Open);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize the pen for drawing an arc
PdfPen pen = new PdfPen(Color.Brown, 10f);
//Set the line join style of the pen
pen.LineCap = PdfLineCap.Square;
//Set the bounds for arc
RectangleF bounds = new RectangleF(20, 40, 200, 200);

```

```
//Draw the arc on PDF document
loadedPage.Graphics.DrawArc(pen, bounds, 270, 90);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownloadName = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize the pen for drawing an arc
PdfPen pen = new PdfPen(Syncfusion.Drawing.Color.Brown, 10f);
//Set the line join style of the pen
pen.LineCap = PdfLineCap.Square;
//Set the bounds for arc
RectangleF bounds = new RectangleF(20, 40, 200, 200);
//Draw the arc on PDF document
loadedPage.Graphics.DrawArc(pen, bounds, 270, 90);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Bezier

You can draw a bezier in PDF document by using the [DrawBezier](#) method of [PdfGraphics](#). The following code snippet explains how to draw a bezier in new PDF document.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize pen to draw the bezier
PdfPen pen = new PdfPen(PdfBrushes.Brown, 1f);
//Draw the bezier on PDF document
page.Graphics.DrawBezier(pen, new PointF(10, 10), new PointF(10, 50), new
PointF(50, 80), new PointF(80, 10));
//Save the PDF document
document.Save("Output.pdf");
//Close the instance of PdfDocument
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = document.Pages.Add
'Initialize pen to draw the bezier
Dim pen As PdfPen = New PdfPen(PdfBrushes.Brown, 1.0F)
'Draw the bezier on PDF document
page.Graphics.DrawBezier(pen, New PointF(10, 10), New PointF(10, 50), New
PointF(50, 80), New PointF(80, 10))
'Save the PDF document
document.Save("Output.pdf")
'Close the instance of PdfDocument
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize pen to draw the bezier
PdfPen pen = new PdfPen(PdfBrushes.Brown, 1f);
//Draw the bezier on PDF document
page.Graphics.DrawBezier(pen, new PointF(10, 10), new PointF(10, 50), new
PointF(50, 80), new PointF(80, 10));
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize pen to draw the bezier
PdfPen pen = new PdfPen(PdfBrushes.Brown, 1f);
//Draw the bezier on PDF document
page.Graphics.DrawBezier(pen, new PointF(10, 10), new PointF(10, 50), new
PointF(50, 80), new PointF(80, 10));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileName = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize pen to draw the bezier
PdfPen pen = new PdfPen(PdfBrushes.Brown, 1f);
//Draw the bezier on PDF document
page.Graphics.DrawBezier(pen, new PointF(10, 10), new PointF(10, 50), new
PointF(50, 80), new PointF(80, 10));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

The following code snippet explains how to draw a bezier in an existing PDF document.

C#

```
//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize pen to draw the bezier
PdfPen pen = new PdfPen(PdfBrushes.Brown, 1f);
//Draw the bezier on PDF document
loadedPage.Graphics.DrawBezier(pen, new PointF(10, 10), new PointF(10, 50),
new PointF(50, 80), new PointF(80, 10));
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
```

VB.NET

```
'Load an existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the page into PdfLoadedPage
Dim loadedPage As PdfLoadedPage = CType(loadedDocument.Pages(0),
PdfLoadedPage)
'Initialize pen to draw the bezier
Dim pen As PdfPen = New PdfPen(PdfBrushes.Brown, 1.0F)
'Draw the bezier on PDF document
loadedPage.Graphics.DrawBezier(pen, New PointF(10, 10), New PointF(10, 50),
New PointF(50, 80), New PointF(80, 10))
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the instance of PdfLoadedDocument
loadedDocument.Close(True)
```

UWP

```
//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize pen to draw the bezier
PdfPen pen = new PdfPen(PdfBrushes.Brown, 1f);
//Draw the bezier on PDF document
loadedPage.Graphics.DrawBezier(pen, new PointF(10, 10), new PointF(10, 50),
new PointF(50, 80), new PointF(80, 10));
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document as stream
FileStream inputStream = new FileStream("Input.pdf", FileMode.Open);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize pen to draw the bezier
PdfPen pen = new PdfPen(PdfBrushes.Brown, 1f);
//Draw the bezier on PDF document
loadedPage.Graphics.DrawBezier(pen, new PointF(10, 10), new PointF(10, 50),
new PointF(50, 80), new PointF(80, 10));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize pen to draw the bezier
PdfPen pen = new PdfPen(PdfBrushes.Brown, 1f);
//Draw the bezier on PDF document
loadedPage.Graphics.DrawBezier(pen, new PointF(10, 10), new PointF(10, 50),
new PointF(50, 80), new PointF(80, 10));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```



```
}
```

Ellipse

You can draw an ellipse in PDF document by using the [DrawEllipse](#) method of [PdfGraphics](#). The following code snippet explains how to draw an ellipse in new PDF document.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfSolidBrush for drawing the ellipse
PdfSolidBrush brush = new PdfSolidBrush(Color.Red);
//Draw ellipse on the page
page.Graphics.DrawEllipse(brush, new RectangleF(10, 10, 200, 100));
//Save the PDF document
document.Save("Output.pdf");
//Close the instance of PdfDocument
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Add a page to the document
Dim page As PdfPage = document.Pages.Add
'Initialize PdfSolidBrush for drawing the ellipse
Dim brush As PdfSolidBrush = New PdfSolidBrush(Color.Red)
'Draw ellipse on the page
page.Graphics.DrawEllipse(brush, New RectangleF(10, 10, 200, 100))
'Save the PDF document
document.Save("Output.pdf")
'Close the instance of PdfDocument
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfSolidBrush for drawing the ellipse
PdfSolidBrush brush = new PdfSolidBrush(Color.FromArgb(255, 255, 0, 0));
//Draw ellipse on the page
page.Graphics.DrawEllipse(brush, new RectangleF(10, 10, 200, 100));
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfSolidBrush for drawing the ellipse
PdfSolidBrush brush = new PdfSolidBrush(Color.Red);
//Draw ellipse on the page
page.Graphics.DrawEllipse(brush, new RectangleF(10, 10, 200, 100));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileName = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page to the document
PdfPage page = document.Pages.Add();
//Initialize PdfSolidBrush for drawing the ellipse
PdfSolidBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Red);
//Draw ellipse on the page
page.Graphics.DrawEllipse(brush, new RectangleF(10, 10, 200, 100));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the instance of PdfDocument
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

The following code snippet explains how to draw an ellipse in an existing PDF document.

C#

```
//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize PdfSolidBrush for drawing the ellipse
PdfSolidBrush brush = new PdfSolidBrush(Color.Red);
//Draw ellipse on the page
loadedPage.Graphics.DrawEllipse(brush, new RectangleF(10, 10, 200, 100));
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
```

VB.NET

```
'Load an existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the page into PdfLoadedPage
Dim loadedPage As PdfLoadedPage = CType(loadedDocument.Pages(0),
PdfLoadedPage)
'Initialize PdfSolidBrush for drawing the ellipse
Dim brush As PdfSolidBrush = New PdfSolidBrush(Color.Red)
'Draw ellipse on the page
loadedPage.Graphics.DrawEllipse(brush, New RectangleF(10, 10, 200, 100))
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the instance of PdfLoadedDocument
loadedDocument.Close(True)
```

UWP

```
//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize PdfSolidBrush for drawing the ellipse
PdfSolidBrush brush = new PdfSolidBrush(Color.FromArgb(255, 255, 0, 0));
//Draw ellipse on the page
loadedPage.Graphics.DrawEllipse(brush, new RectangleF(10, 10, 200, 100));
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document as stream
FileStream inputStream = new FileStream("Input.pdf", FileMode.Open);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize PdfSolidBrush for drawing the ellipse
PdfSolidBrush brush = new PdfSolidBrush(Color.Red);
//Draw ellipse on the page
loadedPage.Graphics.DrawEllipse(brush, new RectangleF(10, 10, 200, 100));
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Load the PDF document as stream
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page into PdfLoadedPage
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Initialize PdfSolidBrush for drawing the ellipse
PdfSolidBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Red);
//Draw ellipse on the page
loadedPage.Graphics.DrawEllipse(brush, new RectangleF(10, 10, 200, 100));
//Save the document to the stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Working with shape pagination

You can also allow large shapes to paginate across pages by assigning `Paginate` of `PdfLayoutType` Enum to `Layout` property of `PdfLayoutFormat` class.

C#

```
//Create Document
PdfDocument doc = new PdfDocument();
//Add new page
PdfPage page = doc.Pages.Add();
//Set bounds for ellipse
RectangleF rect = new RectangleF(0, 0, 100, 1000);
//Create ellipse
PdfEllipse ellipse = new PdfEllipse(rect);
//Set layout property to make the ellipse break across the pages.
PdfLayoutFormat format = new PdfLayoutFormat();
format.Break = PdfLayoutBreakType.FitPage;
format.Layout = PdfLayoutType.Paginate;
ellipse.Brush = PdfBrushes.Brown;
//Draw ellipse.
ellipse.Draw(page, 20, 20, format);
//Save and close the PDF
doc.Save("Shapes.pdf");
doc.Close(true);
```

VB.NET

```
'Create Document
Dim doc As New PdfDocument()
'Add new page
Dim page As PdfPage = doc.Pages.Add()
'Set bounds for ellipse
Dim rect As New RectangleF(0, 0, 100, 1000)
'Create ellipse
Dim ellipse As New PdfEllipse(rect)
'Set layout property to make the ellipse break across the pages.
Dim format As New PdfLayoutFormat()
format.Break = PdfLayoutBreakType.FitPage
format.Layout = PdfLayoutType.Paginate
ellipse.Brush = PdfBrushes.Brown
'Draw ellipse.
ellipse.Draw(page, 20, 20, format)
'Save and close the PDF
doc.Save("Shapes.pdf")
doc.Close(True)
```

UWP

```
//Create Document
PdfDocument doc = new PdfDocument();
//Add new page
PdfPage page = doc.Pages.Add();
//Set bounds for ellipse
RectangleF rect = new RectangleF(0, 0, 100, 1000);
//Create ellipse
PdfEllipse ellipse = new PdfEllipse(rect);
```

```
//Set layout property to make the ellipse break across the pages.
PdfLayoutFormat format = new PdfLayoutFormat();
format.Break = PdfLayoutBreakType.FitPage;
format.Layout = PdfLayoutType.Paginate;
ellipse.Brush = PdfBrushes.Brown;
//Draw ellipse.
ellipse.Draw(page, 20, 20, format);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Shapes.pdf");
```

ASP.NET CORE

```
//Create Document
PdfDocument doc = new PdfDocument();
//Add new page
PdfPage page = doc.Pages.Add();
//Set bounds for ellipse
RectangleF rect = new RectangleF(0, 0, 100, 1000);
//Create ellipse
PdfEllipse ellipse = new PdfEllipse(rect);
//Set layout property to make the ellipse break across the pages.
PdfLayoutFormat format = new PdfLayoutFormat();
format.Break = PdfLayoutBreakType.FitPage;
format.Layout = PdfLayoutType.Paginate;
ellipse.Brush = PdfBrushes.Brown;
//Draw ellipse.
ellipse.Draw(page, 20, 20, format);
//Save the document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
stream.Position = 0;
//Closes the document
doc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Shapes.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create Document
PdfDocument doc = new PdfDocument();
//Add new page
PdfPage page = doc.Pages.Add();
//Set bounds for ellipse
RectangleF rect = new RectangleF(0, 0, 100, 1000);
//Create ellipse
```

```

PdfEllipse ellipse = new PdfEllipse(rect);
//Set layout property to make the ellipse break across the pages.
PdfLayoutFormat format = new PdfLayoutFormat();
format.Break = PdfLayoutBreakType.FitPage;
format.Layout = PdfLayoutType.Paginate;
ellipse.Brush = PdfBrushes.Brown;
//Draw ellipse.
ellipse.Draw(page, 20, 20, format);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Closes the document
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Shapes.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Shapes.pdf",
"application/pdf", stream);
}

```

Working with Bookmarks

Essential PDF provides support to insert, remove and modify the bookmarks in the PDF Document.

Adding Bookmarks in a PDF

The [PdfBookmarkBase](#) collection represents the bookmarks in a PDF document. You can add a bookmark in a new PDF document using [PdfBookmark](#) class. Please refer the following code example.

C#

```

//Creates a new document.
PdfDocument document = new PdfDocument();
//Adds a page.
PdfPage page = document.Pages.Add();
//Creates document bookmarks.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Sets the destination page.
bookmark.Destination = new PdfDestination(page);
//Sets the destination location.
bookmark.Destination.Location = new PointF(20, 20);
//Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Color.Red;
//Saves and closes the PDF document.
document.Save("Output.pdf");
document.Close(True);

```

VB.NET

```

'Creates a new document.
Dim document As New PdfDocument()
'Adds a page.
Dim page As PdfPage = document.Pages.Add()
'Creates document bookmarks.
Dim bookmark As PdfBookmark = document.Bookmarks.Add("Page 1")
'Sets the destination page.
bookmark.Destination = New PdfDestination(page)
'Sets the destination location.
bookmark.Destination.Location = New PointF(20, 20)
'Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold
bookmark.Color = Color.Red
'Saves and closes the PDF document.
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Creates a new document.
PdfDocument document = new PdfDocument();
//Adds a page.
PdfPage page = document.Pages.Add();
//Creates document bookmarks.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Sets the destination page.
bookmark.Destination = new PdfDestination(page);
//Sets the destination location.
bookmark.Destination.Location = new PointF(20, 20);
//Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Color.FromArgb(0, 255, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Creates a new document.
PdfDocument document = new PdfDocument();
//Adds a page.
PdfPage page = document.Pages.Add();
//Creates document bookmarks.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Sets the destination page.
bookmark.Destination = new PdfDestination(page);
//Sets the destination location.
bookmark.Destination.Location = new PointF(20, 20);
//Sets the text style and color.

```



```

bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Syncfusion.Drawing.Color.Red;
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates a new document.
PdfDocument document = new PdfDocument();
//Adds a page.
PdfPage page = document.Pages.Add();
//Creates document bookmarks.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Sets the destination page.
bookmark.Destination = new PdfDestination(page);
//Sets the destination location.
bookmark.Destination.Location = new PointF(20, 20);
//Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Syncfusion.Drawing.Color.Red;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Adding Bookmarks in an existing PDF document

To add bookmarks to an existing PDF document, use the following code example.

C#

```
//Loads the document.
PdfLoadedDocument document = new PdfLoadedDocument("input.pdf");
//Creates document bookmarks.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Sets the destination page.
bookmark.Destination = new PdfDestination(document.Pages[0]);
//Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Color.Red;
//Sets the destination location.
bookmark.Destination.Location = new PointF(20, 20);
//Saves and closes the PDF document.
document.Save("Output.pdf");
document.Close(True);
```

VB.NET

```
'Loads the document.
Dim document As New PdfLoadedDocument("input.pdf")
'Creates document bookmarks.
Dim bookmark As PdfBookmark = document.Bookmarks.Add("Page 1")
'Sets the destination page.
bookmark.Destination = New PdfDestination(document.Pages(0))
'Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold
bookmark.Color = Color.Red
'Sets the destination location.
bookmark.Destination.Location = New PointF(20, 20)
'Saves and closes the PDF document.
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await document.OpenAsync(file);
//Creates document bookmarks.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Sets the destination page.
bookmark.Destination = new PdfDestination(document.Pages[0]);
//Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Color.FromArgb(0, 255, 0, 0);
//Sets the destination location.
bookmark.Destination.Location = new PointF(20, 20);
//Save the PDF document to stream
```

```

MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Creates document bookmarks.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Sets the destination page.
bookmark.Destination = new PdfDestination(document.Pages[0]);
//Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Color.Red;
//Sets the destination location.
bookmark.Destination.Location = new PointF(20, 20);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Creates document bookmarks.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Sets the destination page.
bookmark.Destination = new PdfDestination(document.Pages[0]);
//Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Syncfusion.Drawing.Color.Red;
//Sets the destination location.
bookmark.Destination.Location = new PointF(20, 20);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);

```

```

//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Adding a Child to the Bookmarks

You can add a child bookmark by using [Insert](#) method. Please refer to the following code example.

C#

```

//Creates a new document.
PdfDocument document = new PdfDocument();
//Adds a page.
PdfPage page = document.Pages.Add();
//Creates bookmark.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Sets the destination page.
bookmark.Destination = new PdfDestination(page);
//Sets the destination location.
bookmark.Destination.Location = new PointF(20, 20);
//Adds the child bookmark
PdfBookmark childBookmark = bookmark.Insert(0, "heading 1");
childBookmark.Destination = new PdfDestination(page);
childBookmark.Destination.Location = new PointF(400, 300);
childBookmark.Destination.Zoom = 2F;
//Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Color.Red;
//Saves and closes the PDF document.
document.Save("Output.pdf");
document.Close(True);

```

VB.NET

```

'Creates a new document.
Dim document As New PdfDocument()
'Adds a page.
Dim page As PdfPage = document.Pages.Add()
'Creates bookmark.
Dim bookmark As PdfBookmark = document.Bookmarks.Add("Page 1")
'Sets the destination page.
bookmark.Destination = New PdfDestination(page)
'Sets the destination location.

```

```

bookmark.Destination.Location = New PointF(20, 20)
'Adds the child bookmark
Dim childBookmark As PdfBookmark = bookmark.Insert(0, "heading 1")
childBookmark.Destination = New PdfDestination(page)
childBookmark.Destination.Location = New PointF(400, 300)
childBookmark.Destination.Zoom = 2.0F
'Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold
bookmark.Color = Color.Red
'Saves and closes the PDF document.
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Creates a new document.
PdfDocument document = new PdfDocument();
//Adds a page.
PdfPage page = document.Pages.Add();
//Creates bookmark.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Sets the destination page.
bookmark.Destination = new PdfDestination(page);
//Sets the destination location.
bookmark.Destination.Location = new PointF(20, 20);
//Adds the child bookmark
PdfBookmark childBookmark = bookmark.Insert(0, "heading 1");
childBookmark.Destination = new PdfDestination(page);
childBookmark.Destination.Location = new PointF(400, 300);
childBookmark.Destination.Zoom = 2F;
//Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Color.FromArgb(0, 255, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Creates a new document.
PdfDocument document = new PdfDocument();
//Adds a page.
PdfPage page = document.Pages.Add();
//Creates bookmark.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Sets the destination page.
bookmark.Destination = new PdfDestination(page);
//Sets the destination location.
bookmark.Destination.Location = new PointF(20, 20);
//Adds the child bookmark
PdfBookmark childBookmark = bookmark.Insert(0, "heading 1");

```

```

childBookmark.Destination = new PdfDestination(page);
childBookmark.Destination.Location = new PointF(400, 300);
childBookmark.Destination.Zoom = 2F;
//Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Syncfusion.Drawing.Color.Red;
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates a new document.
PdfDocument document = new PdfDocument();
//Adds a page.
PdfPage page = document.Pages.Add();
//Creates bookmark.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Sets the destination page.
bookmark.Destination = new PdfDestination(page);
//Sets the destination location.
bookmark.Destination.Location = new PointF(20, 20);
//Adds the child bookmark
PdfBookmark childBookmark = bookmark.Insert(0, "heading 1");
childBookmark.Destination = new PdfDestination(page);
childBookmark.Destination.Location = new PointF(400, 300);
childBookmark.Destination.Zoom = 2F;
//Sets the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Syncfusion.Drawing.Color.Red;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else

```

```
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}
```

Inserting Bookmarks in an existing PDF

When loading an existing document, the Essential PDF loads all bookmarks of the document.

Each loaded bookmark is represented by the [PdfLoadedBookmark](#) object. The following code example illustrates how to insert new bookmarks in the existing PDF document.

C#

```
//Creates a new document.
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//Inserts a new bookmark in the existing bookmark collection.
PdfBookmark bookmark = document.Bookmarks.Insert(1, "New Page 2");
//Sets the destination page and location.
bookmark.Destination = new PdfDestination(document.Pages[1]);
bookmark.Destination.Location = new PointF(0, 300);
//Saves and closes the PDF document.
document.Save("Output.pdf");
document.Close(True);
```

VB.NET

```
'Creates a new document.
Dim document As New PdfLoadedDocument("Input.pdf")
'Inserts a new bookmark in the existing bookmark collection.
Dim bookmark As PdfBookmark = document.Bookmarks.Insert(1, "New Page 2")
'Sets the destination page and location.
bookmark.Destination = New PdfDestination(document.Pages(1))
bookmark.Destination.Location = New PointF(0, 300)
'Saves and closes the PDF document.
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await document.OpenAsync(file);
//Inserts a new bookmark in the existing bookmark collection.
PdfBookmark bookmark = document.Bookmarks.Insert(1, "New Page 2");
//Sets the destination page and location.
bookmark.Destination = new PdfDestination(document.Pages[1]);
bookmark.Destination.Location = new PointF(0, 300);
//Save the PDF document to stream
```

```

MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Inserts a new bookmark in the existing bookmark collection.
PdfBookmark bookmark = document.Bookmarks.Insert(1, "New Page 2");
//Sets the destination page and location.
bookmark.Destination = new PdfDestination(document.Pages[1]);
bookmark.Destination.Location = new PointF(0, 300);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Inserts a new bookmark in the existing bookmark collection.
PdfBookmark bookmark = document.Bookmarks.Insert(1, "New Page 2");
//Sets the destination page and location.
bookmark.Destination = new PdfDestination(document.Pages[1]);
bookmark.Destination.Location = new PointF(0, 300);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)

```



```
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Removing Bookmarks from an existing PDF

You can also remove bookmarks from the existing PDF document by using [Remove](#) method. Please refer the following code example.

C#

```
//Loads the PDF document.
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//Gets all the bookmarks.
PdfBookmarkBase bookmarks = document.Bookmarks;
//Removes bookmark by bookmark name.
bookmarks.Remove("Page 1");
//Removes bookmark by index.
bookmarks.RemoveAt(1);
//Saves and closes the document.
document.Save("Output.pdf");
document.Close(True);
```

VB.NET

```
'Loads the PDF document.
Dim document As New PdfLoadedDocument("Input.pdf")
'Gets all the bookmarks.
Dim bookmarks As PdfBookmarkBase = document.Bookmarks
'Removes bookmark by bookmark name.
bookmarks.Remove("Page 1")
'Removes bookmark by index.
bookmarks.RemoveAt(1)
'Saves and closes the document.
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await document.OpenAsync(file);
//Gets all the bookmarks.
```

```

PdfBookmarkBase bookmarks = document.Bookmarks;
//Removes bookmark by bookmark name.
bookmarks.Remove("Page 1");
//Removes bookmark by index.
bookmarks.RemoveAt(1);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Gets all the bookmarks.
PdfBookmarkBase bookmarks = document.Bookmarks;
//Removes bookmark by bookmark name.
bookmarks.Remove("Page 1");
//Removes bookmark by index.
bookmarks.RemoveAt(1);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Gets all the bookmarks.
PdfBookmarkBase bookmarks = document.Bookmarks;
//Removes bookmark by bookmark name.
bookmarks.Remove("Page 1");
//Removes bookmark by index.
bookmarks.RemoveAt(1);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);

```

```
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Modifying the Bookmarks

Essential PDF allows you to modify the bookmarks in the existing PDF document. The following modifications can be done to bookmarks in an existing document.

- Modify the bookmark style, color, title, and destination.
- Add or insert new bookmarks into the root collection.
- Add or insert new bookmarks as a child of another bookmark.
- Assign the destination of the added bookmarks to a loaded page or a new page of the document.

The following code example shows how to modify the [Destination](#), [Color](#), [TextStyle](#) and [Title](#) of an existing bookmark collection.

C#

```
//Loads the PDF document.
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//Gets all the bookmarks.
PdfBookmarkBase bookmarks = document.Bookmarks;
//Gets the first bookmark and changes the properties of the bookmark.
PdfLoadedBookmark bookmark = bookmarks[0] as PdfLoadedBookmark;
bookmark.Destination = new PdfDestination(document.Pages[1]);
bookmark.Color = Color.Green;
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Title = "Changed title";
//Saves the document
document.Save("Output.pdf");
document.Close(True);
```

VB.NET

```
'Loads the PDF document.
Dim document As New PdfLoadedDocument("Input.pdf")
'Gets all the bookmarks.
Dim bookmarks As PdfBookmarkBase = document.Bookmarks
'Gets the first bookmark and changes the properties of the bookmark.
```

```

Dim bookmark As PdfLoadedBookmark = TryCast(bookmarks(0), PdfLoadedBookmark)
bookmark.Destination = New PdfDestination(document.Pages(1))
bookmark.Color = Color.Green
bookmark.TextStyle = PdfTextStyle.Bold
bookmark.Title = "Changed title"
'Saves the document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await document.OpenAsync(file);
//Gets all the bookmarks.
PdfBookmarkBase bookmarks = document.Bookmarks;
//Gets the first bookmark and changes the properties of the bookmark.
PdfLoadedBookmark bookmark = bookmarks[0] as PdfLoadedBookmark;
bookmark.Destination = new PdfDestination(document.Pages[1]);
bookmark.Color = Color.FromArgb(0, 0, 128, 0);
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Title = "Changed title";
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open, FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Gets all the bookmarks.
PdfBookmarkBase bookmarks = document.Bookmarks;
//Gets the first bookmark and changes the properties of the bookmark.
PdfLoadedBookmark bookmark = bookmarks[0] as PdfLoadedBookmark;
bookmark.Destination = new PdfDestination(document.Pages[0]);
bookmark.Color = Syncfusion.Drawing.Color.Green;
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Title = "Changed title";
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;

```

```
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Gets all the bookmarks.
PdfBookmarkBase bookmarks = document.Bookmarks;
//Gets the first bookmark and changes the properties of the bookmark.
PdfLoadedBookmark bookmark = bookmarks[0] as PdfLoadedBookmark;
bookmark.Destination = new PdfDestination(document.Pages[0]);
bookmark.Color = Syncfusion.Drawing.Color.Green;
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Title = "Changed title";
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Working with Named Destination

Essential PDF provides support to add, remove and modify the named destination in the PDF document. When you open a PDF file in a web browser, the first page of the PDF file will be shown by default. By adding a named destination, you can open the PDF with the desired location and magnification. The following link example shows how to open a PDF document with named destination in a web page.

Another example that uses "nameddest" parameter in URL:

e.g. <http://www.syncfusion.com/downloads/support/directtrac/general/pd/mydocument-1524150305.pdf#nameddest=Chapter3>

Points to remembers:

- Individual parameters, together with their values (separated by & or #), can be no greater than 32 characters in length.
- You cannot use the reserved characters =, #, and &. There is no way to escape these special characters.

Adding Named Destination to a PDF document

You can add, remove and modify the named destination using [PdfNamedDestination](#) class.

The following code example shows how to add named destination in a new PDF document.

C#

```
//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page to the document.
PdfPage page = doc.Pages.Add();
//Create an instance for named destination.
PdfNamedDestination destination = new PdfNamedDestination("TOC");
destination.Destination = new PdfDestination(page);
//Set the location
destination.Destination.Location = new PointF(0, 500);
//Set zoom factor to 400 percentage
destination.Destination.Zoom = 4;
doc.NamedDestinationCollection.Add(destination);
//Draw the text.
page.Graphics.DrawString("Hello World!!", new
PdfStandardFont(PdfFontFamily.Helvetica, 10), PdfBrushes.Black, new
PointF(0, 500));
//Save the document.
doc.Save("Output.pdf");
//Close the document.
doc.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page to the document.
Dim page As PdfPage = doc.Pages.Add()
'Create an instance for named destination.
Dim destination As New PdfNamedDestination("TOC")
destination.Destination = New PdfDestination(page)
'Set the location
destination.Destination.Location = New PointF(0, 500)
'Set zoom factor to 400 percentage
destination.Destination.Zoom = 4
doc.NamedDestinationCollection.Add(destination)
'Draw the text.
```

```

page.Graphics.DrawString("Hello World!!", New
PdfStandardFont(PdfFontFamily.Helvetica, 10), PdfBrushes.Black, New
PointF(0, 500))
'Save the document.
doc.Save("Output.pdf")
'Close the document.
doc.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page to the document.
PdfPage page = doc.Pages.Add();
//Create an instance for named destination.
PdfNamedDestination destination = new PdfNamedDestination("TOC");
destination.Destination = new PdfDestination(page);
//Set the location
destination.Destination.Location = new PointF(0, 500);
//Set zoom factor to 400 percentage
destination.Destination.Zoom = 4;
doc.NamedDestinationCollection.Add(destination);
//Draw the text.
page.Graphics.DrawString("Hello World!!", new
PdfStandardFont(PdfFontFamily.Helvetica, 10), PdfBrushes.Black, new
PointF(0, 500));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page to the document.
PdfPage page = doc.Pages.Add();
//Create an instance for named destination.
PdfNamedDestination destination = new PdfNamedDestination("TOC");
destination.Destination = new PdfDestination(page);
//Set the location
destination.Destination.Location = new PointF(0, 500);
//Set zoom factor to 400 percentage
destination.Destination.Zoom = 4;
doc.NamedDestinationCollection.Add(destination);
//Draw the text.
page.Graphics.DrawString("Hello World!!", new
PdfStandardFont(PdfFontFamily.Helvetica, 10), PdfBrushes.Black, new
PointF(0, 500));
//Save the document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);

```

```

stream.Position = 0;
//Closes the document
doc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page to the document.
PdfPage page = doc.Pages.Add();
//Create an instance for named destination.
PdfNamedDestination destination = new PdfNamedDestination("TOC");
destination.Destination = new PdfDestination(page);
//Set the location
destination.Destination.Location = new PointF(0, 500);
//Set zoom factor to 400 percentage
destination.Destination.Zoom = 4;
doc.NamedDestinationCollection.Add(destination);
//Draw the text.
page.Graphics.DrawString("Hello World!!", new
PdfStandardFont(PdfFontFamily.Helvetica, 10), PdfBrushes.Black, new
PointF(0, 500));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Closes the document
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Adding Named Destination to an existing PDF document

The following code example shows how to add named destination in an existing PDF document using the [PdfNamedDestination](#) class.

C#


```

//Load the PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get the first page of the document
PdfPageBase page = loadedDocument.Pages[0];
//Create an instance for named destination.
PdfNamedDestination destination = new PdfNamedDestination("TOC");
destination.Destination = new PdfDestination(page);
//Set the location
destination.Destination.Location = new PointF(0, 500);
//Set zoom factor to 400 percentage
destination.Destination.Zoom = 4;
loadedDocument.NamedDestinationCollection.Add(destination);
//Save the document
loadedDocument.Save("Output.pdf");
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Get the first page of the document
Dim page As PdfPageBase = loadedDocument.Pages(0)
'Create an instance for named destination.
Dim destination As New PdfNamedDestination("TOC")
destination.Destination = New PdfDestination(page)
'Set the location
destination.Destination.Location = New PointF(0, 500)
'Set zoom factor to 400 percentage
destination.Destination.Zoom = 4
loadedDocument.NamedDestinationCollection.Add(destination)
'Save the document
loadedDocument.Save("Output.pdf")
'Close the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the first page of the document
PdfPageBase page = loadedDocument.Pages[0];
//Create an instance for named destination.
PdfNamedDestination destination = new PdfNamedDestination("TOC");
destination.Destination = new PdfDestination(page);
//Set the location
destination.Destination.Location = new PointF(0, 500);
//Set zoom factor to 400 percentage

```

```

destination.Destination.Zoom = 4;
loadedDocument.NamedDestinationCollection.Add(destination);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the first page of the document
PdfPageBase page = loadedDocument.Pages[0];
//Create an instance for named destination.
PdfNamedDestination destination = new PdfNamedDestination("TOC");
destination.Destination = new PdfDestination(page);
//Set the location
destination.Destination.Location = new PointF(0, 500);
//Set zoom factor to 400 percentage
destination.Destination.Zoom = 4;
loadedDocument.NamedDestinationCollection.Add(destination);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Closes the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the first page of the document
PdfPageBase page = loadedDocument.Pages[0];
//Create an instance for named destination.
PdfNamedDestination destination = new PdfNamedDestination("TOC");
destination.Destination = new PdfDestination(page);
//Set the location
destination.Destination.Location = new PointF(0, 500);
//Set zoom factor to 400 percentage

```

```

destination.Destination.Zoom = 4;
loadedDocument.NamedDestinationCollection.Add(destination);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Closes the document
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Removing/Modifying the named destination

You can remove the named destination using [Remove](#) method of [PdfNamedDestinationCollection](#). The following code snippet illustrates the same.

C#

```

//Load the PDF document
PdfLoadedDocument lDoc = new PdfLoadedDocument("Sample.pdf");
//Get the named destination collection
PdfNamedDestinationCollection destinationCollection =
lDoc.NamedDestinationCollection;
//Remove the named destination by title
destinationCollection.Remove("TOC");
//Modify the exiting named destination
PdfNamedDestination destination = destinationCollection[0];
destination.Title = "POC";
//Save the document
lDoc.Save("Output.pdf");
//Close the document
lDoc.Close(true);

```

VB.NET

```

'Load the PDF document
Dim lDoc As New PdfLoadedDocument("Sample.pdf")
'Get the named destination collection
Dim destinationCollection As PdfNamedDestinationCollection =
lDoc.NamedDestinationCollection
'Remove the named destination by title
destinationCollection.Remove("TOC")
'Modify the exiting named destination
Dim destination As PdfNamedDestination = destinationCollection(0)

```

```
destination.Title = "POC"
'Save the document
lDoc.Save("Output.pdf")
'Close the document
lDoc.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await lDoc.OpenAsync(file);
//Get the named destination collection
PdfNamedDestinationCollection destinationCollection =
lDoc.NamedDestinationCollection;
//Remove the named destination by title
destinationCollection.Remove("TOC");
//Modify the exiting named destination
PdfNamedDestination destination = destinationCollection[0];
destination.Title = "POC";
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await lDoc.SaveAsync(stream);
//Close the document
lDoc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Barcode.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Get the named destination collection
PdfNamedDestinationCollection destinationCollection =
lDoc.NamedDestinationCollection;
//Remove the named destination by title
destinationCollection.Remove("TOC");
//Modify the exiting named destination
PdfNamedDestination destination = destinationCollection[0];
destination.Title = "POC";
//Save the document into stream.
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
stream.Position = 0;
//Close the documents.
lDoc.Close(true);
//Defining the ContentType for pdf file.
```

```

string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Get the named destination collection
PdfNamedDestinationCollection destinationCollection =
lDoc.NamedDestinationCollection;
//Remove the named destination by title
destinationCollection.Remove("TOC");
//Modify the exiting named destination
PdfNamedDestination destination = destinationCollection[0];
destination.Title = "POC";
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
//Closes the document
lDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Adding named destination to the bookmarks

The following code example shows how to add named destination to the [Bookmarks](#) in the PDF document.

C#

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page to the document.
PdfPage page = doc.Pages.Add();
//Create an instance for named destination.
PdfNamedDestination destination = new PdfNamedDestination("TOC");

```

```

destination.Destination = new PdfDestination(page);
//Set the location
destination.Destination.Location = new PointF(0, 800);
//Set zoom factor to 400 percentage
destination.Destination.Zoom = 4;
//Add the named destination to the collection
doc.NamedDestinationCollection.Add(destination);
//Create a bookmark
PdfBookmark bookmark = doc.Bookmarks.Add("TOC");
//Assign the named destination to the bookmark
bookmark.NamedDestination = destination;
//Save the document
doc.Save("Sample.pdf");
//Close the document
doc.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim doc As New PdfDocument()
'Add a page to the document.
Dim page As PdfPage = doc.Pages.Add()
'Create an instance for named destination.
Dim destination As New PdfNamedDestination("TOC")
destination.Destination = New PdfDestination(page)
'Set the location
destination.Destination.Location = New PointF(0, 800)
'Set zoom factor to 400 percentage
destination.Destination.Zoom = 4
'Add the named destination to the collection
doc.NamedDestinationCollection.Add(destination)
'Create a bookmark
Dim bookmark As PdfBookmark = doc.Bookmarks.Add("TOC")
'Assign the named destination to the bookmark
bookmark.NamedDestination = destination
'Save the document
doc.Save("Sample.pdf")
'Close the document
doc.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page to the document.
PdfPage page = doc.Pages.Add();
//Create an instance for named destination.
PdfNamedDestination destination = new PdfNamedDestination("TOC");
destination.Destination = new PdfDestination(page);
//Set the location
destination.Destination.Location = new PointF(0, 800);
//Set zoom factor to 400 percentage
destination.Destination.Zoom = 4;
//Add the named destination to the collection
doc.NamedDestinationCollection.Add(destination);
//Create a bookmark

```

```

PdfBookmark bookmark = doc.Bookmarks.Add("TOC");
//Assign the named destination to the bookmark
bookmark.NamedDestination = destination;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page to the document.
PdfPage page = doc.Pages.Add();
//Create an instance for named destination.
PdfNamedDestination destination = new PdfNamedDestination("TOC");
destination.Destination = new PdfDestination(page);
//Set the location
destination.Destination.Location = new PointF(0, 800);
//Set zoom factor to 400 percentage
destination.Destination.Zoom = 4;
//Add the named destination to the collection
doc.NamedDestinationCollection.Add(destination);
//Create a bookmark
PdfBookmark bookmark = doc.Bookmarks.Add("TOC");
//Assign the named destination to the bookmark
bookmark.NamedDestination = destination;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
doc.Save(stream);
stream.Position = 0;
//Close the documents.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument doc = new PdfDocument();
//Add a page to the document.
PdfPage page = doc.Pages.Add();
//Create an instance for named destination.
PdfNamedDestination destination = new PdfNamedDestination("TOC");
destination.Destination = new PdfDestination(page);
//Set the location
destination.Destination.Location = new PointF(0, 800);

```

```

//Set zoom factor to 400 percentage
destination.Destination.Zoom = 4;
//Add the named destination to the collection
doc.NamedDestinationCollection.Add(destination);
//Create a bookmark
PdfBookmark bookmark = doc.Bookmarks.Add("TOC");
//Assign the named destination to the bookmark
bookmark.NamedDestination = destination;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Closes the document
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

Working with Annotations

Essential PDF provides support for interactive annotations.

You can add, delete and modify the annotation from the PDF documents.

Adding annotations to a PDF document

You can add a popup annotation to the page using [PdfPopupAnnotation](#) class. The following code example explains this.

C#

```

//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new popup annotation.
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test
popup annotation");
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the pdf popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds this annotation to the created page.
page.Annotations.Add(popupAnnotation);

```



```
//Saves the document to disk.
document.Save("PopupAnnotation.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document.
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Creates a new rectangle
Dim rectangle As New RectangleF(10, 40, 30, 30)
'Creates a new popup annotation.
Dim popupAnnotation As New PdfPopupAnnotation(rectangle, "Test popup
annotation")
popupAnnotation.Border.Width = 4
popupAnnotation.Border.HorizontalRadius = 20
popupAnnotation.Border.VerticalRadius = 30
'Sets the pdf popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph
'Adds this annotation to the created page.
page.Annotations.Add(popupAnnotation)
'Saves the document to disk.
document.Save("PopupAnnotation.pdf")
document.Close(True)
```

UWP

```
//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new popup annotation.
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test
popup annotation");
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the pdf popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds this annotation to the created page.
page.Annotations.Add(popupAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "PopupAnnotation.pdf");
```

ASP.NET CORE

```
//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new popup annotation.
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test
popup annotation");
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the pdf popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds this annotation to the created page.
page.Annotations.Add(popupAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "PopupAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new popup annotation.
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation();
popupAnnotation.Bounds = rectangle;
popupAnnotation.Text = "Test popup annotation";
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the pdf popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds this annotation to the created page.
page.Annotations.Add(popupAnnotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("PopupAnnotation.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("PopupAnnotation.pdf",
"application/pdf", stream);
}
```

To add annotations to an existing PDF document, use the following code example.

C#

```
//Creates a new PDF document.
PdfLoadedDocument document = new PdfLoadedDocument("input.pdf");
//Creates a rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new popup annotation.
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test
popup annotation");
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the pdf popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds the annotation to loaded page
document.Pages[0].Annotations.Add(popupAnnotation);
//Saves the document to disk.
document.Save("PopupAnnotation.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document.
Dim document As New PdfLoadedDocument("input.pdf")
'Creates a rectangle
Dim rectangle As New RectangleF(10, 40, 30, 30)
'Creates a new popup annotation.
Dim popupAnnotation As New PdfPopupAnnotation(rectangle, "Test popup
annotation")
popupAnnotation.Border.Width = 4
popupAnnotation.Border.HorizontalRadius = 20
popupAnnotation.Border.VerticalRadius = 30
'Sets the pdf popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph
'Adds the annotation to loaded page
document.Pages(0).Annotations.Add(popupAnnotation)
'Saves the document to disk.
document.Save("PopupAnnotation.pdf")
document.Close(True)
```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await document.OpenAsync(file);
//Creates a rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new popup annotation.
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test popup annotation");
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the pdf popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds the annotation to loaded page
document.Pages[0].Annotations.Add(popupAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples
Save(stream, "PopupAnnotation.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open, FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Creates a rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new popup annotation.
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test popup annotation");
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the pdf popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds the annotation to loaded page
document.Pages[0].Annotations.Add(popupAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;

```

```
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "PopupAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Creates a rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new popup annotation.
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test
popup annotation");
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the pdf popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds the annotation to loaded page
document.Pages[0].Annotations.Add(popupAnnotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("PopupAnnotati
on.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("PopupAnnotation.pdf",
"application/pdf", stream);
}
```

Flatten annotation

Annotations can be flattened by removing the existing annotation and replacing it with graphics objects that would resemble the annotation and cannot be edited.

This can be achieved by enabling the [Flatten](#) property. Please refer the sample for flattening all the annotations in the PDF document.

C#

```
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("input.pdf");
//Get all the pages
foreach (PdfLoadedPage loadedPage in loadedDocument.Pages)
{
    //Flatten all the annotations in the page
    loadedPage.Annotations.Flatten = true;
}
//Save and close the PDF document instance
loadedDocument.Save("output.pdf");
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document
Dim loadedDocument As New PdfLoadedDocument("input.pdf")
'Get all the pages
For Each loadedPage As PdfLoadedPage In loadedDocument.Pages
    'Flatten all the annotations in the page
    loadedPage.Annotations.Flatten = True
Next
'Save and close the PDF document instance
loadedDocument.Save("output.pdf")
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get all the pages
foreach (PdfLoadedPage loadedPage in loadedDocument.Pages)
{
    //Flatten all the annotations in the page
    loadedPage.Annotations.Flatten = true;
}
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get all the pages
foreach (PdfLoadedPage loadedPage in loadedDocument.Pages)
{
//Flatten all the annotations in the page
loadedPage.Annotations.Flatten = true;
}
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Closes the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get all the pages
foreach (PdfLoadedPage loadedPage in loadedDocument.Pages)
{
//Flatten all the annotations in the page
loadedPage.Annotations.Flatten = true;
}
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
"application/pdf", stream);
}
else
```

```
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
"application/pdf", stream);
}
```

To flatten the specific annotation in the PDF document, use the below code example.

C#

```
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("input.pdf");
//Get all the pages
foreach (PdfLoadedPage loadedPage in loadedDocument.Pages)
{
//Flatten all the annotations in the page
foreach (PdfLoadedAnnotation annotation in loadedPage.Annotations)
{
//Check for the circle annotation
if (annotation is PdfLoadedCircleAnnotation)
{
//Flatten the circle annotation
annotation.Flatten = true;
}
}
}
//Save and close the PDF document instance
loadedDocument.Save("Output.pdf");
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document
Dim loadedDocument As New PdfLoadedDocument("input.pdf")
'Get all the pages
For Each loadedPage As PdfLoadedPage In loadedDocument.Pages
'Flatten all the annotations in the page
For Each annotation As PdfLoadedAnnotation In loadedPage.Annotations
'Check for the circle annotation
If TypeOf annotation Is PdfLoadedCircleAnnotation Then
'Flatten the circle annotation
annotation.Flatten = True
End If
Next
Next
'Save and close the PDF document instance
loadedDocument.Save("Output.pdf")
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
```



```

//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get all the pages
foreach (PdfLoadedPage loadedPage in loadedDocument.Pages)
{
    //Flatten all the annotations in the page
    foreach (PdfLoadedAnnotation annotation in loadedPage.Annotations)
    {
        //Check for the circle annotation
        if (annotation is PdfLoadedCircleAnnotation)
        {
            //Flatten the circle annotation
            annotation.Flatten = true;
        }
    }
}
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open, FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get all the pages
foreach (PdfLoadedPage loadedPage in loadedDocument.Pages)
{
    //Flatten all the annotations in the page
    foreach (PdfLoadedAnnotation annotation in loadedPage.Annotations)
    {
        //Check for the circle annotation
        if (annotation is PdfLoadedCircleAnnotation)
        {
            //Flatten the circle annotation
            annotation.Flatten = true;
        }
    }
}
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Closes the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";

```

```
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get all the pages
foreach (PdfLoadedPage loadedPage in loadedDocument.Pages)
{
    //Flatten all the annotations in the page
    foreach (PdfLoadedAnnotation annotation in loadedPage.Annotations)
    {
        //Check for the circle annotation
        if (annotation is PdfLoadedCircleAnnotation)
        {
            //Flatten the circle annotation
            annotation.Flatten = true;
        }
    }
}
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
"application/pdf", stream);
}
```

To flatten pop-up annotation in the PDF document, use the following code example.

C#

```
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new
PdfLoadedDocument("PopupAnnotation.pdf");
//Get all the pages
```

```

foreach(PdfLoadedPage loadedPage in loadedDocument.Pages)
{
    foreach(PdfLoadedAnnotation annotation in loadedPage.Annotations)
    {
        if(annotation is PdfLoadedPopupAnnotation)
        {
            //Enable the flatten annotation
            annotation.Flatten = true;
            //Enable flatten for the pop-up window annotation
            annotation.FlattenPopUps = true;
        }
    }
}
//Save the document
loadedDocument.Save("Output.pdf");
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the existing PDF document
Dim loadedDocument As New PdfLoadedDocument("PopupAnnotation.pdf")
'Get all the pages
For Each loadedPage As PdfLoadedPage In loadedDocument.Pages
For Each annotation As PdfLoadedAnnotation In loadedPage.Annotations
If TypeOf annotation Is PdfLoadedPopupAnnotation Then
'Enable the flatten annotation
annotation.Flatten = True
'Enable flatten for the pop-up window annotation
annotation.FlattenPopUps = True
End If
Next
Next
'Save the document
loadedDocument.Save("Output.pdf")
'Close the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document using Open method of the
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get all the page
foreach (PdfLoadedPage loadedPage in loadedDocument.Pages)
{
    foreach (PdfLoadedAnnotation annotation in loadedPage.Annotations)
    {
        if (annotation is PdfLoadedPopupAnnotation)

```

```

{
    //Enable the flatten annotation
    annotation.Flatten = true;
    //Enable flatten for the pop-up window annotation
    annotation.FlattenPopUps = true;
}
}
}
//Save the document as stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document instances
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("PopupAnnotation.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get all the pages
foreach (PdfLoadedPage loadedPage in loadedDocument.Pages)
{
    foreach (PdfLoadedAnnotation annotation in loadedPage.Annotations)
    {
        if (annotation is PdfLoadedPopupAnnotation)
        {
            //Enable the flatten annotation
            annotation.Flatten = true;
            //Enable flatten for the pop-up window annotation
            annotation.FlattenPopUps = true;
        }
    }
}
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
loadedDocument.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates the FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream

```

```

Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
PopupAnnotation.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get all the pages
foreach (PdfLoadedPage loadedPage in loadedDocument.Pages)
{
    foreach (PdfLoadedAnnotation annotation in loadedPage.Annotations)
    {
        if (annotation is PdfLoadedPopupAnnotation)
        {
            //Enable the flatten annotation
            annotation.Flatten = true;
            //Enable flatten for the pop-up window annotation
            annotation.FlattenPopUps = true;
        }
    }
}
//Save the document as stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document instances
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Supported annotation types

3D Annotation

3D Annotations are used to represent 3D artworks in a PDF document. Essential PDF provides support to embed 3D files (u3d) in PDF.

You can add a 3D annotation in PDF document using [Pdf3DAnnotation](#) class. The following example illustrates this.

C#

```

//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new pdf 3d annotation.
Pdf3DAnnotation pdf3dAnnotation = new Pdf3DAnnotation(new RectangleF(10, 50,
300, 150), @"3DAnnotation.U3D");

```

```
//Handles the activation of the 3d annotation
Pdf3DActivation activation = new Pdf3DActivation();
activation.ActivationMode = Pdf3DActivationMode.ExplicitActivation;
activation.ShowToolbar = true;
pdf3dAnnotation.Activation = activation;
//Adds annotation to page
page.Annotations.Add(pdf3dAnnotation);
//Saves the document to disk.
document.Save("3DAnnotation.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document.
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Creates a new pdf 3d annotation.
Dim pdf3dAnnotation As New Pdf3DAnnotation(New RectangleF(10, 50, 300, 150),
"3DAnnotation.U3D")
'Handles the activation of the 3d annotation
Dim activation As New Pdf3DActivation()
activation.ActivationMode = Pdf3DActivationMode.ExplicitActivation
activation.ShowToolbar = True
pdf3dAnnotation.Activation = activation
'Adds annotation to page
page.Annotations.Add(pdf3dAnnotation)
'Saves the document to disk.
document.Save("3DAnnotation.pdf")
document.Close(True)
```

UWP

```
//PDF supports 3D annotation only in Windows Forms, WPF, ASP.NET, ASP.NET
MVC and ASP.NET Core.
```

ASP.NET CORE

```
//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
FileStream inputStream = new FileStream("3DAnnotation.U3D", FileMode.Open,
FileAccess.Read);
//Creates a new pdf 3d annotation.
Pdf3DAnnotation pdf3dAnnotation = new Pdf3DAnnotation(new RectangleF(10, 50,
300, 150), inputStream);
//Handles the activation of the 3d annotation
Pdf3DActivation activation = new Pdf3DActivation();
activation.ActivationMode = Pdf3DActivationMode.ExplicitActivation;
activation.ShowToolbar = true;
pdf3dAnnotation.Activation = activation;
//Adds annotation to page
page.Annotations.Add(pdf3dAnnotation);
//Save the document into stream
```

```

MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "3DAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.3DAnnotation.u3d");
//Creates a new PDF 3d annotation
Pdf3DAnnotation pdf3dAnnotation = new Pdf3DAnnotation(new RectangleF(10, 50,
300, 150), inputStream);
//Handles the activation of the 3d annotation
Pdf3DActivation activation = new Pdf3DActivation();
activation.ActivationMode = Pdf3DActivationMode.ExplicitActivation;
activation.ShowToolbar = true;
pdf3dAnnotation.Activation = activation;
//Adds annotation to page
page.Annotations.Add(pdf3dAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("3DAnnotation.
pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("3DAnnotation.pdf",
"application/pdf", stream);
}

```

You can add the JavaScript script to the 3D annotation using the [OnInstantiate](#) property, which is executed whenever a 3D stream is read to create an instance of the 3D artwork. The following code snippet illustrate this.

C#

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Create a new PDF 3D annotation
Pdf3DAnnotation pdf3dAnnotation = new Pdf3DAnnotation(new RectangleF(10, 50,
300, 150), @"Input.u3d");
//Assign JavaScript script
pdf3dAnnotation.OnInstantiate = "host.getURL(\"http://www.google.com\")";
//Adds annotation to page
page.Annotations.Add(pdf3dAnnotation);
//Save the document to disk
document.Save("3DAnnotation.pdf");
//Close the document
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Create a new PDF 3D annotation
Dim pdf3dAnnotation As New Pdf3DAnnotation(New RectangleF(10, 50, 300, 150),
"Input.u3d")
'Assign JavaScript script
pdf3dAnnotation.OnInstantiate = "host.getURL(\"http://www.google.com\")"
'Adds annotation to page
page.Annotations.Add(pdf3dAnnotation)
'Save the document to disk
document.Save("3DAnnotation.pdf")
'Close the document
document.Close(True)
```

UWP

```
//PDF supports 3D annotation only in Windows Forms, WPF, ASP.NET, ASP.NET
MVC, and ASP.NET Core platforms
```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
FileStream inputStream = new FileStream("3DAnnotation.U3D", FileMode.Open,
FileAccess.Read);
//Creates a new PDF 3D annotation
Pdf3DAnnotation pdf3dAnnotation = new Pdf3DAnnotation(new RectangleF(10, 50,
300, 150), inputStream);
//Assign JavaScript script
pdf3dAnnotation.OnInstantiate = "host.getURL(\"http://www.google.com\")";
//Adds annotation to page
```



```

page.Annotations.Add(pdf3dAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "3DAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.3DAnnotation.u3d");
//Creates a new PDF 3d annotation
Pdf3DAnnotation pdf3dAnnotation = new Pdf3DAnnotation(new RectangleF(10, 50,
300, 150), inputStream);
//Assign JavaScript script
pdf3dAnnotation.OnInstantiate = "host.getURL(\"http://www.google.com\")";
//Adds annotation to page
page.Annotations.Add(pdf3dAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("3DAnnotation.
pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("3DAnnotation.pdf",
"application/pdf", stream);
}

```

File Link Annotation

Links for external files can be added in a PDF document by using the [PdfFileLinkAnnotation](#) class.

The following code example explains how to add a file link annotation in PDF.

C#

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new pdf file link annotation.
PdfFileLinkAnnotation fileLinkAnnotation = new
PdfFileLinkAnnotation(rectangle, @"logo.png");
//Adds this annotation to a new page.
page.Annotations.Add(fileLinkAnnotation);
//Saves the document to disk.
document.Save("FileLinkAnnotation.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Creates a new rectangle
Dim rectangle As New RectangleF(10, 40, 30, 30)
'Creates a new pdf file link annotation.
Dim fileLinkAnnotation As New PdfFileLinkAnnotation(rectangle, "logo.png")
'Adds this annotation to a new page.
page.Annotations.Add(fileLinkAnnotation)
'Saves the document to disk.
document.Save("FileLinkAnnotation.pdf")
document.Close(True)
```

UWP

```
//PDF supports File Link Annotation only in Windows Forms, WPF, ASP.NET and
ASP.NET MVC.
```

ASP.NET CORE

```
//PDF supports File Link Annotation only in Windows Forms, WPF, ASP.NET and
ASP.NET MVC.
```

XAMARIN

```
//PDF supports File Link Annotation only in Windows Forms, WPF, ASP.NET and
ASP.NET MVC.
```

Free Text Annotation

Free text annotation enables you to display the text directly on the page. When you want to add a comment directly without placing it on a pop-up window, [PdfFreeTextAnnotation](#) can be used.

The following code example explains how to add a free text annotation in the PDF document.

C#

```

//Creates a new pdf document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates PDF free text annotation
PdfFreeTextAnnotation freeText = new PdfFreeTextAnnotation(new
RectangleF(50, 100, 100, 50));
//Sets properties to the annotation
freeText.MarkupText = "Free Text with Callout";
freeText.TextMarkupColor = new PdfColor(Color.Black);
freeText.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 7f);
freeText.Color = new PdfColor(Color.Yellow);
freeText.BorderColor = new PdfColor(Color.Red);
freeText.Border = new PdfAnnotationBorder(.5f);
freeText.LineEndingStyle = PdfLineEndingStyle.OpenArrow;
freeText.AnnotationFlags = PdfAnnotationFlags.Default;
freeText.Text = "Free Text";
freeText.Opacity = 0.5f;
PointF[] points = { new PointF(100, 450), new PointF(100, 200), new
PointF(100, 150) };
freeText.CalloutLines = points;
//Adds the annotation to page
page.Annotations.Add(freeText);
//Saves the document to disk.
document.Save("FreeTextAnnotation.pdf");
document.Close(true);

```

VB.NET

```

'Creates a new pdf document
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Creates PDF free text annotation
Dim freeText As New PdfFreeTextAnnotation(New RectangleF(50, 100, 100, 50))
'Sets properties to the annotation
freeText.MarkupText = "Free Text with Callout"
freeText.TextMarkupColor = New PdfColor(Color.Black)
freeText.Font = New PdfStandardFont(PdfFontFamily.Helvetica, 7.0F)
freeText.Color = New PdfColor(Color.Yellow)
freeText.BorderColor = New PdfColor(Color.Red)
freeText.Border = New PdfAnnotationBorder(0.5F)
freeText.LineEndingStyle = PdfLineEndingStyle.OpenArrow
freeText.AnnotationFlags = PdfAnnotationFlags.[Default]
freeText.Text = "Free Text"
freeText.Opacity = 0.5F
Dim points As PointF() = { New PointF(100, 450), New PointF(100, 200), New
PointF(100, 150) }
freeText.CalloutLines = points
'Adds the annotation to page
page.Annotations.Add(freeText)
'Saves the document to disk.
document.Save("FreeTextAnnotation.pdf")
document.Close(True)

```

UWP

```

//Creates a new pdf document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates PDF free text annotation
PdfFreeTextAnnotation freeText = new PdfFreeTextAnnotation(new
RectangleF(50, 100, 100, 50));
//Sets properties to the annotation
freeText.MarkupText = "Free Text with Callout";
freeText.TextMarkupColor = new PdfColor(0, 0, 0);
freeText.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 7f);
freeText.Color = new PdfColor(255, 255, 0);
freeText.BorderColor = new PdfColor(255, 0, 0);
freeText.Border = new PdfAnnotationBorder(.5f);
freeText.LineEndingStyle = PdfLineEndingStyle.OpenArrow;
freeText.AnnotationFlags = PdfAnnotationFlags.Default;
freeText.Text = "Free Text";
freeText.Opacity = 0.5f;
PointF[] points = { new PointF(100, 450), new PointF(100, 200), new
PointF(100, 150) };
freeText.CalloutLines = points;
//Adds the annotation to page
page.Annotations.Add(freeText);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respect
Save(stream, "FreeTextAnnotation.pdf");

```

ASP.NET CORE

```

//Creates a new pdf document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates PDF free text annotation
PdfFreeTextAnnotation freeText = new PdfFreeTextAnnotation(new
RectangleF(50, 100, 100, 50));
//Sets properties to the annotation
freeText.MarkupText = "Free Text with Callout";
freeText.TextMarkupColor = new PdfColor(Color.Black);
freeText.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 7f);
freeText.Color = new PdfColor(Color.Yellow);
freeText.BorderColor = new PdfColor(Color.Red);
freeText.Border = new PdfAnnotationBorder(.5f);
freeText.LineEndingStyle = PdfLineEndingStyle.OpenArrow;
freeText.AnnotationFlags = PdfAnnotationFlags.Default;
freeText.Text = "Free Text";
freeText.Opacity = 0.5f;

```

```

PointF[] points = { new PointF(100, 450), new PointF(100, 200), new
PointF(100, 150) };
freeText.CalloutLines = points;
//Adds the annotation to page
page.Annotations.Add(freeText);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "FreeTextAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates a new pdf document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates PDF free text annotation
PdfFreeTextAnnotation freeText = new PdfFreeTextAnnotation(new
RectangleF(50, 100, 100, 50));
//Sets properties to the annotation
freeText.MarkupText = "Free Text with Callout";
freeText.TextMarkupColor = new PdfColor(Syncfusion.Drawing.Color.Black);
freeText.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 7f);
freeText.Color = new PdfColor(Syncfusion.Drawing.Color.Yellow);
freeText.BorderColor = new PdfColor(Syncfusion.Drawing.Color.Red);
freeText.Border = new PdfAnnotationBorder(.5f);
freeText.LineEndingStyle = PdfLineEndingStyle.OpenArrow;
freeText.AnnotationFlags = PdfAnnotationFlags.Default;
freeText.Text = "Free Text";
freeText.Opacity = 0.5f;
PointF[] points = { new PointF(100, 450), new PointF(100, 200), new
PointF(100, 150) };
freeText.CalloutLines = points;
//Adds the annotation to page
page.Annotations.Add(freeText);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("FreeTextAnnotation.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("FreeTextAnnotation.pdf", "application/pdf", stream);
}
```

Line Annotation

Line annotation displays a single straight line on the page. When you open it, it displays a pop-up window containing text of the associated note.

[PdfLineAnnotation](#) is used to create and set the properties of the Line annotation.

C#

```
//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Specifies the line end points
int[] points = new int[] { 80, 420, 150, 420 };
//Creates a new line annotation.
PdfLineAnnotation lineAnnotation = new PdfLineAnnotation(points, "Line Annotation");
//Creates pdf line border
LineBorder lineBorder = new LineBorder();
lineBorder.BorderStyle = PdfBorderStyle.Solid;
lineBorder.BorderWidth = 1;
lineAnnotation.LineBorder = lineBorder;
lineAnnotation.LineIntent = PdfLineIntent.LineDimension;
//Assigns the line ending style
lineAnnotation.BeginLineStyle = PdfLineEndingStyle.Butt;
lineAnnotation.EndLineStyle = PdfLineEndingStyle.Diamond;
lineAnnotation.AnnotationFlags = PdfAnnotationFlags.Default;
//Assigns the line color
lineAnnotation.InnerLineColor = new PdfColor(Color.Green);
lineAnnotation.BackColor = new PdfColor(Color.Green);
//Assigns the leader line
lineAnnotation.LeaderLineExt = 0;
lineAnnotation.LeaderLine = 0;
//Assigns the Line caption type
lineAnnotation.LineCaption = true;
lineAnnotation.CaptionType = PdfLineCaptionType.Inline;
//Adds this annotation to a new page.
page.Annotations.Add(lineAnnotation);
//Saves the document to disk.
document.Save("LineAnnotation.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document.
Dim document As New PdfDocument()
'Creates a new page
```

```

Dim page As PdfPage = document.Pages.Add()
'Specifies the line end points
Dim points As Integer() = New Integer() {80, 420, 150, 420}
'Creates a new line annotation.
Dim lineAnnotation As New PdfLineAnnotation(points, "Line Annotation")
'Creates pdf line border
Dim lineBorder As New LineBorder()
lineBorder.BorderStyle = PdfBorderStyle.Solid
lineBorder.BorderWidth = 1
lineAnnotation.lineBorder = lineBorder
lineAnnotation.LineIntent = PdfLineIntent.LineDimension
'Assigns the line ending style
lineAnnotation.BeginLineStyle = PdfLineEndingStyle.Butt
lineAnnotation.EndLineStyle = PdfLineEndingStyle.Diamond
lineAnnotation.AnnotationFlags = PdfAnnotationFlags.[Default]
'Assigns the line color
lineAnnotation.InnerLineColor = New PdfColor(Color.Green)
lineAnnotation.BackColor = New PdfColor(Color.Green)
'Assigns the leader line
lineAnnotation.LeaderLineExt = 0
lineAnnotation.LeaderLine = 0
'Assigns the Line caption type
lineAnnotation.LineCaption = True
lineAnnotation.CaptionType = PdfLineCaptionType.Inline
'Adds this annotation to a new page.
page.Annotations.Add(lineAnnotation)
'Saves the document to disk.
document.Save("LineAnnotation.pdf")
document.Close(True)

```

UWP

```

//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Specifies the line end points
int[] points = new int[] { 80, 420, 150, 420 };
//Creates a new line annotation.
PdfLineAnnotation lineAnnotation = new PdfLineAnnotation(points, "Line
Annotation");
//Creates pdf line border
LineBorder lineBorder = new LineBorder();
lineBorder.BorderStyle = PdfBorderStyle.Solid;
lineBorder.BorderWidth = 1;
lineAnnotation.lineBorder = lineBorder;
lineAnnotation.LineIntent = PdfLineIntent.LineDimension;
//Assigns the line ending style
lineAnnotation.BeginLineStyle = PdfLineEndingStyle.Butt;
lineAnnotation.EndLineStyle = PdfLineEndingStyle.Diamond;
lineAnnotation.AnnotationFlags = PdfAnnotationFlags.Default;
//Assigns the line color
lineAnnotation.InnerLineColor = new PdfColor(0, 128, 0);
lineAnnotation.BackColor = new PdfColor(0, 128, 0);
//Assigns the leader line
lineAnnotation.LeaderLineExt = 0;

```

```

lineAnnotation.LeaderLine = 0;
//Assigns the Line caption type
lineAnnotation.LineCaption = true;
lineAnnotation.CaptionType = PdfLineCaptionType.Inline;
//Adds this annotation to a new page.
page.Annotations.Add(lineAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "LineAnnotation.pdf");

```

ASP.NET CORE

```

//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Specifies the line end points
int[] points = new int[] { 80, 420, 150, 420 };
//Creates a new line annotation.
PdfLineAnnotation lineAnnotation = new PdfLineAnnotation(points, "Line
Annotation");
//Creates pdf line border
LineBorder lineBorder = new LineBorder();
lineBorder.BorderStyle = PdfBorderStyle.Solid;
lineBorder.BorderWidth = 1;
lineAnnotation.lineBorder = lineBorder;
lineAnnotation.LineIntent = PdfLineIntent.LineDimension;
//Assigns the line ending style
lineAnnotation.BeginLineStyle = PdfLineEndingStyle.Butt;
lineAnnotation.EndLineStyle = PdfLineEndingStyle.Diamond;
lineAnnotation.AnnotationFlags = PdfAnnotationFlags.Default;
//Assigns the line color
lineAnnotation.InnerLineColor = new PdfColor(Color.Green);
lineAnnotation.BackColor = new PdfColor(Color.Green);
//Assigns the leader line
lineAnnotation.LeaderLineExt = 0;
lineAnnotation.LeaderLine = 0;
//Assigns the Line caption type
lineAnnotation.LineCaption = true;
lineAnnotation.CaptionType = PdfLineCaptionType.Inline;
//Adds this annotation to a new page.
page.Annotations.Add(lineAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name

```



```
string fileName = "LineAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Specifies the line end points
int[] points = new int[] { 80, 420, 150, 420 };
//Creates a new line annotation.
PdfLineAnnotation lineAnnotation = new PdfLineAnnotation(points, "Line
Annotation");
//Creates pdf line border
LineBorder lineBorder = new LineBorder();
lineBorder.BorderStyle = PdfBorderStyle.Solid;
lineBorder.BorderWidth = 1;
lineAnnotation.LineBorder = lineBorder;
lineAnnotation.LineIntent = PdfLineIntent.LineDimension;
//Assigns the line ending style
lineAnnotation.BeginLineStyle = PdfLineEndingStyle.Butt;
lineAnnotation.EndLineStyle = PdfLineEndingStyle.Diamond;
lineAnnotation.AnnotationFlags = PdfAnnotationFlags.Default;
//Assigns the line color
lineAnnotation.InnerLineColor = new
PdfColor(Syncfusion.Drawing.Color.Green);
lineAnnotation.BackColor = new PdfColor(Syncfusion.Drawing.Color.Green);
//Assigns the leader line
lineAnnotation.LeaderLineExt = 0;
lineAnnotation.LeaderLine = 0;
//Assigns the Line caption type
lineAnnotation.LineCaption = true;
lineAnnotation.CaptionType = PdfLineCaptionType.Inline;
//Adds this annotation to a new page.
page.Annotations.Add(lineAnnotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("LineAnnotatio
n.pdf", "application/pdf", stream);
}
else
{
```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("LineAnnotation.pdf",
"application/pdf", stream);
}
```

Rubber stamp Annotation

Rubber stamp annotation displays text or graphics intended to look like it is stamped on the page with a rubber stamp.

When opened, it displays a pop-up window containing the text of the associated note.

[PdfRubberStampAnnotation](#) is used to create rubber stamp annotation.

C#

```
//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new pdf rubber stamp annotation.
RectangleF rectangle = new RectangleF(40, 60, 80, 20);
PdfRubberStampAnnotation rubberStampAnnotation = new
PdfRubberStampAnnotation(rectangle, " Text Rubber Stamp Annotation");
rubberStampAnnotation.Icon = PdfRubberStampAnnotationIcon.Draft;
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation";
//Adds annotation to the page
page.Annotations.Add(rubberStampAnnotation);
//Saves the document to disk.
document.Save("RubberStamp.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document.
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Creates a new pdf rubber stamp annotation.
Dim rectangle As New RectangleF(40, 60, 80, 20)
Dim rubberStampAnnotation As New PdfRubberStampAnnotation(rectangle, " Text
Rubber Stamp Annotation")
rubberStampAnnotation.Icon = PdfRubberStampAnnotationIcon.Draft
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation"
'Adds annotation to the page
page.Annotations.Add(rubberStampAnnotation)
'Saves the document to disk.
document.Save("RubberStamp.pdf")
document.Close(True)
```

UWP

```
//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new pdf rubber stamp annotation.
RectangleF rectangle = new RectangleF(40, 60, 80, 20);
```

```

PdfRubberStampAnnotation rubberStampAnnotation = new
PdfRubberStampAnnotation(rectangle, " Text Rubber Stamp
rubberStampAnnotation.Icon = PdfRubberStampAnnotationIcon.Draft;
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation";
//Adds annotation to the page
page.Annotations.Add(rubberStampAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "RubberStamp.pdf");

```

ASP.NET CORE

```

//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new pdf rubber stamp annotation.
RectangleF rectangle = new RectangleF(40, 60, 80, 20);
PdfRubberStampAnnotation rubberStampAnnotation = new
PdfRubberStampAnnotation(rectangle, " Text Rubber Stamp Annotation");
rubberStampAnnotation.Icon = PdfRubberStampAnnotationIcon.Draft;
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation";
//Adds annotation to the page
page.Annotations.Add(rubberStampAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "RubberStamp.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new pdf rubber stamp annotation.
RectangleF rectangle = new RectangleF(40, 60, 80, 20);
PdfRubberStampAnnotation rubberStampAnnotation = new
PdfRubberStampAnnotation(rectangle, " Text Rubber Stamp Annotation");
rubberStampAnnotation.Icon = PdfRubberStampAnnotationIcon.Draft;
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation";
//Adds annotation to the page

```

```

page.Annotations.Add(rubberStampAnnotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("RubberStamp.p
df", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("RubberStamp.pdf",
"application/pdf", stream);
}

```

Ink Annotation

Ink annotation represents freehand “scribble” comprising one or more disjoint paths.

When you open it, it displays a pop-up window containing text of the associated note.

[PdfInkAnnotation](#) is used to create ink annotation in a PDF document.

C#

```

//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
List<float> linePoints=new List<float>{40,300,60,100,40,50,40,300};
//Creates a new ink annotation
RectangleF rectangle = new RectangleF(0, 0, 300, 400);
PdfInkAnnotation inkAnnotation = new PdfInkAnnotation(rectangle,linePoints);
inkAnnotation.Color = new PdfColor(Color.Red);
//Adds annotation to the page
page.Annotations.Add(inkAnnotation);
//Saves the document to disk.
document.Save("InkAnnotation.pdf");
document.Close(true);

```

VB.NET

```

'Creates a new PDF document.
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
Dim linePoints As New List(Of Single)() From {40, 300, 60, 100, 40, 50, 40,
300}
'Creates a new ink annotation
Dim rectangle As New RectangleF(0, 0, 300, 400)

```

```

Dim inkAnnotation As New PdfInkAnnotation(rectangle, linePoints)
inkAnnotation.Color = New PdfColor(Color.Red)
'Adds annotation to the page
page.Annotations.Add(inkAnnotation)
'Saves the document to disk.
document.Save("InkAnnotation.pdf")
document.Close(True)

```

UWP

```

//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
List<float> linePoints = new List<float> { 40, 300, 60, 100, 40, 50, 40, 300
};
//Creates a new ink annotation
RectangleF rectangle = new RectangleF(0, 0, 300, 400);
PdfInkAnnotation inkAnnotation = new PdfInkAnnotation(rectangle,
linePoints);
inkAnnotation.Color = new PdfColor(Color.FromArgb(0,255,0,0));
//Adds annotation to the page
page.Annotations.Add(inkAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "InkAnnotation.pdf");

```

ASP.NET CORE

```

//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
List<float> linePoints = new List<float> { 40, 300, 60, 100, 40, 50, 40, 300
};
//Creates a new ink annotation
RectangleF rectangle = new RectangleF(0, 0, 300, 400);
PdfInkAnnotation inkAnnotation = new PdfInkAnnotation(rectangle,
linePoints);
inkAnnotation.Color = new PdfColor(Color.Red);
//Adds annotation to the page
page.Annotations.Add(inkAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name

```

```
string fileName = "InkAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
List<float> linePoints = new List<float> { 40, 300, 60, 100, 40, 50, 40, 300
};
//Creates a new ink annotation
RectangleF rectangle = new RectangleF(0, 0, 300, 400);
PdfInkAnnotation inkAnnotation = new PdfInkAnnotation(rectangle,
linePoints);
inkAnnotation.Color = new PdfColor(Syncfusion.Drawing.Color.Red);
//Adds annotation to the page
page.Annotations.Add(inkAnnotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("InkAnnotation
.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("InkAnnotation.pdf",
"application/pdf", stream);
}
```

You can get ink list points from the [PdfLoadedInkAnnotation](#), represented by [InkPointsCollection](#). The following code illustrate this.

C#

```
//Loads the document
PdfLoadedDocument lDoc = new PdfLoadedDocument("Input.pdf");
//Gets the first page from the document
PdfLoadedPage page = lDoc.Pages[0] as PdfLoadedPage;
//Gets the annotation collection
PdfLoadedAnnotationCollection annotations = page.Annotations;
//Gets the first ink annotation
PdfLoadedInkAnnotation inkAnnotation = annotations[0] as
PdfLoadedInkAnnotation;
//Gets the ink points collection
```

```
List<List<float>> points = inkAnnotation.InkPointsCollection;
//Save the document
lDoc.Save("Output.pdf");
//Close the document
lDoc.Close(true);
```

VB.NET

```
'Loads the document
Dim lDoc As New PdfLoadedDocument("Input.pdf")
'Gets the first page from the document
Dim page As PdfLoadedPage = TryCast(lDoc.Pages(0), PdfLoadedPage)
'Gets the annotation collection
Dim annotations As PdfLoadedAnnotationCollection = page.Annotations
'Gets the first ink annotation
Dim inkAnnotation As PdfLoadedInkAnnotation = TryCast(annotations(0), PdfLoadedInkAnnotation)
'Gets the ink points collection
Dim points As List(Of List(Of Single)) = inkAnnotation.InkPointsCollection
'Save the document
lDoc.Save("Output.pdf")
'Close the document
lDoc.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document using the Open method of PdfLoadedDocument class
await lDoc.OpenAsync(file);
//Gets the first page from the document
PdfLoadedPage page = lDoc.Pages[0] as PdfLoadedPage;
//Gets the annotation collection
PdfLoadedAnnotationCollection annotations = page.Annotations;
//Gets the first ink annotation
PdfLoadedInkAnnotation inkAnnotation = annotations[0] as PdfLoadedInkAnnotation;
//Gets the ink points collection
List<List<float>> points = inkAnnotation.InkPointsCollection;
//Save the document as stream
MemoryStream stream = new MemoryStream();
await lDoc.SaveAsync(stream);
//Close the document instances
lDoc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Gets the first page from the document
PdfLoadedPage page = lDoc.Pages[0] as PdfLoadedPage;
//Gets the annotation collection
PdfLoadedAnnotationCollection annotations = page.Annotations;
//Gets the first ink annotation
PdfLoadedInkAnnotation inkAnnotation = annotations[0] as
PdfLoadedInkAnnotation;
//Gets the ink points collection
List<List<float>> points = inkAnnotation.InkPointsCollection;
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
lDoc.Save(stream);
//If the position is not set to '0', the PDF will be empty.
stream.Position = 0;
//Close the document
lDoc.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Gets the first page from the document
PdfLoadedPage page = lDoc.Pages[0] as PdfLoadedPage;
//Gets the annotation collection
PdfLoadedAnnotationCollection annotations = page.Annotations;
//Gets the first ink annotation
PdfLoadedInkAnnotation inkAnnotation = annotations[0] as
PdfLoadedInkAnnotation;
//Gets the ink points collection
List<List<float>> points = inkAnnotation.InkPointsCollection;
//Save the document as stream
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
//Close the document instances
lDoc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
```



```
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Pop-up Annotation

Pop-up annotation displays text in a pop-up window for entry and editing.

It typically does not appear alone, but is associated with markup annotation, its parent annotation.

[PdfPopupAnnotation](#) is used to add pop-up annotation in a PDF document.

C#

```
//Creates a new rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new popup annotation.
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test
popup annotation");
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the PDF popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds this annotation to a new page.
page.Annotations.Add(popupAnnotation);
//Saves the document to disk.
document.Save("PopupAnnotation.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document.
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Creates a new rectangle
Dim rectangle As New RectangleF(10, 40, 30, 30)
'Creates a new popup annotation.
Dim popupAnnotation As New PdfPopupAnnotation(rectangle, "Test popup
annotation")
popupAnnotation.Border.Width = 4
popupAnnotation.Border.HorizontalRadius = 20
popupAnnotation.Border.VerticalRadius = 30
'Sets the PDF popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph
'Adds this annotation to a new page.
page.Annotations.Add(popupAnnotation)
'Saves the document to disk.
document.Save("PopupAnnotation.pdf")
document.Close(True)
```

UWP

```

//Create new PDF document
PdfDocument document = new PdfDocument();
//Create a new PDF page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new popup annotation.
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test
popup annotation");
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the PDF popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds this annotation to a new page.
page.Annotations.Add(popupAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected c
Save(stream, "PopupAnnotation.pdf");

```

ASP.NET CORE

```

//Create new PDF document
PdfDocument document = new PdfDocument();
//Create a new PDF page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new popup annotation.
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test
popup annotation");
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the PDF popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds this annotation to a new page.
page.Annotations.Add(popupAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "PopupAnnotation.pdf";

```

```
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create new PDF document
PdfDocument document = new PdfDocument();
//Create a new PDF page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new popup annotation.
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test
popup annotation");
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the PDF popup icon.
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds this annotation to a new page.
page.Annotations.Add(popupAnnotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("PopupAnnotati
on.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("PopupAnnotation.pdf",
"application/pdf", stream);
}
```

File Attachment Annotation

File attachment annotation contains reference to a file that typically is embedded in the PDF file.

[PdfAttachmentAnnotation](#) is used to add a file attachment annotation in a PDF document.

C#

```
//Creates a new PDF Document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF attachmentRectangle = new RectangleF(10, 40, 30, 30);
```

```
//Creates a new attachment annotation.
PdfAttachmentAnnotation attachmentAnnotation = new
PdfAttachmentAnnotation(attachmentRectangle, @"logo.png");
//Sets the attachment icon to attachment annotation.
attachmentAnnotation.Icon = PdfAttachmentIcon.PushPin;
//Adds this annotation to a new page.
page.Annotations.Add(attachmentAnnotation);
//Saves the document to disk.
document.Save("AttachmentAnnotation.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF Document.
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Creates a new rectangle
Dim attachmentRectangle As New RectangleF(10, 40, 30, 30)
'Creates a new attachment annotation.
Dim attachmentAnnotation As New PdfAttachmentAnnotation(attachmentRectangle,
"logo.png")
'Sets the attachment icon to attachment annotation.
attachmentAnnotation.Icon = PdfAttachmentIcon.PushPin
'Adds this annotation to a new page.
page.Annotations.Add(attachmentAnnotation)
'Saves the document to disk.
document.Save("AttachmentAnnotation.pdf")
document.Close(True)
```

UWP

```
//Creates a new PDF Document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF attachmentRectangle = new RectangleF(10, 40, 30, 30);
//Load the file as stream
Stream inputStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Logo.png");
//Creates a new attachment annotation.
PdfAttachmentAnnotation attachmentAnnotation = new
PdfAttachmentAnnotation(attachmentRectangle, @"logo.png", inputStream);
//Sets the attachment icon to attachment annotation.
attachmentAnnotation.Icon = PdfAttachmentIcon.PushPin;
//Adds this annotation to a new page.
page.Annotations.Add(attachmentAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
```

```
Save(stream, "AttachmentAnnotation.pdf");
```

ASP.NET CORE

```
//Creates a new PDF Document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF attachmentRectangle = new RectangleF(10, 40, 30, 30);
//Load the PDF document
FileStream inputStream = new FileStream("logo.png", FileMode.Open,
FileStream.Read);
//Creates a new attachment annotation.
PdfAttachmentAnnotation attachmentAnnotation = new
PdfAttachmentAnnotation(attachmentRectangle, @"logo.png", inputStream);
//Sets the attachment icon to attachment annotation.
attachmentAnnotation.Icon = PdfAttachmentIcon.PushPin;
//Adds this annotation to a new page.
page.Annotations.Add(attachmentAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "PopupAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF Document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF attachmentRectangle = new RectangleF(10, 40, 30, 30);
//Load the file as stream
Stream inputStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Logo.png");
//Creates a new attachment annotation.
PdfAttachmentAnnotation attachmentAnnotation = new
PdfAttachmentAnnotation(attachmentRectangle, @"logo.png", inputStream);
//Sets the attachment icon to attachment annotation.
attachmentAnnotation.Icon = PdfAttachmentIcon.PushPin;
//Adds this annotation to a new page.
page.Annotations.Add(attachmentAnnotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
```

```
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("AttachmentAnn
otation.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("AttachmentAnnotation.pdf"
, "application/pdf", stream);
}
```

Sound Annotation

Sound annotation is used to play the sound clip in the PDF Document.

The following code example explains how to add a sound annotation in a PDF document using [PdfSoundAnnotation](#).

C#

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new sound annotation.
PdfSoundAnnotation soundAnnotation = new PdfSoundAnnotation(rectangle,
@"startup.wav");
soundAnnotation.Sound.Encoding = PdfSoundEncoding.Signed;
soundAnnotation.Sound.Channels = PdfSoundChannels.Stereo;
soundAnnotation.Sound.Bits = 16;
soundAnnotation.Color = new PdfColor(Color.Red);
//Sets the pdf sound icon.
soundAnnotation.Icon = PdfSoundIcon.Speaker;
//Adds this annotation to a new page.
page.Annotations.Add(soundAnnotation);
//Saves the document to disk.
document.Save("SoundIcon.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Creates a new rectangle
Dim rectangle As New RectangleF(10, 40, 30, 30)
'Creates a new sound annotation.
```

```

Dim soundAnnotation As New PdfSoundAnnotation(rectangle, "startup.wav")
soundAnnotation.Sound.Encoding = PdfSoundEncoding.Signed
soundAnnotation.Sound.Channels = PdfSoundChannels.Stereo
soundAnnotation.Sound.Bits = 16
soundAnnotation.Color = New PdfColor(Color.Red)
'Sets the pdf sound icon.
soundAnnotation.Icon = PdfSoundIcon.Speaker
'Adds this annotation to a new page.
page.Annotations.Add(soundAnnotation)
'Saves the document to disk.
document.Save("SoundIcon.pdf")
document.Close(True)

```

UWP

```

//PDF supports Sound Annotation only in Windows Forms, WPF, ASP.NET, ASP.NET
MVC and ASP.NET Web.

```

ASP.NET CORE

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new sound annotation.
FileStream inputStream = new FileStream("Startup.wav", FileMode.Open,
FileAccess.Read);
PdfSoundAnnotation soundAnnotation = new PdfSoundAnnotation(rectangle,
inputStream);
soundAnnotation.Sound.Encoding = PdfSoundEncoding.Signed;
soundAnnotation.Sound.Channels = PdfSoundChannels.Stereo;
soundAnnotation.Sound.Bits = 16;
soundAnnotation.Color = new PdfColor(Color.Red);
//Sets the pdf sound icon.
soundAnnotation.Icon = PdfSoundIcon.Speaker;
//Adds this annotation to a new page.
page.Annotations.Add(soundAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "SoundIcon.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new sound annotation.
//Load the file as stream
Stream inputStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Signature.Assets.Startup.wav");
PdfSoundAnnotation soundAnnotation = new PdfSoundAnnotation(rectangle,
inputStream);
soundAnnotation.Sound.Encoding = PdfSoundEncoding.Signed;
soundAnnotation.Sound.Channels = PdfSoundChannels.Stereo;
soundAnnotation.Sound.Bits = 16;
soundAnnotation.Color = new PdfColor(Syncfusion.Drawing.Color.Red);
//Sets the pdf sound icon.
soundAnnotation.Icon = PdfSoundIcon.Speaker;
//Adds this annotation to a new page.
page.Annotations.Add(soundAnnotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("SoundIcon.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("SoundIcon.pdf",
"application/pdf", stream);
}

```

URI Annotation

URI annotation is used to navigate to a particular web URI

The following code example explains how to add URI annotation in a PDF document using [PdfUriAnnotation](#).

C#

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);

```



```

//Creates a new Uri Annotation
PdfUriAnnotation uriAnnotation = new PdfUriAnnotation(rectangle,
"http://www.google.com");
//Sets Text to uriAnnotation
uriAnnotation.Text = "Uri Annotation";
//Adds this annotation to a new page
page.Annotations.Add(uriAnnotation);
//Saves the document to disk
document.Save("UriAnnotation.pdf");
document.Close(true);

```

VB.NET

```

'Creates a new PDF document
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Creates a new rectangle
Dim rectangle As New RectangleF(10, 40, 30, 30)
'Creates a new Uri Annotation
Dim uriAnnotation As New PdfUriAnnotation(rectangle,
"http://www.google.com")
'Sets Text to uriAnnotation.
uriAnnotation.Text = "Uri Annotation"
'Adds this annotation to a new page
page.Annotations.Add(uriAnnotation)
'Saves the document to disk.
document.Save("UriAnnotation.pdf")
document.Close(True)

```

UWP

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new Uri Annotation
PdfUriAnnotation uriAnnotation = new PdfUriAnnotation(rectangle,
"http://www.google.com");
//Sets Text to uriAnnotation
uriAnnotation.Text = "Uri Annotation";
//Adds this annotation to a new page
page.Annotations.Add(uriAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respective code sample.
Save(stream, "UriAnnotation.pdf");

```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new Uri Annotation
PdfUriAnnotation uriAnnotation = new PdfUriAnnotation(rectangle,
"http://www.google.com");
//Sets Text to uriAnnotation
uriAnnotation.Text = "Uri Annotation";
//Adds this annotation to a new page
page.Annotations.Add(uriAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "UriAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new rectangle
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
//Creates a new Uri Annotation
PdfUriAnnotation uriAnnotation = new PdfUriAnnotation(rectangle,
"http://www.google.com");
//Sets Text to uriAnnotation
uriAnnotation.Text = "Uri Annotation";
//Adds this annotation to a new page
page.Annotations.Add(uriAnnotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("UriAnnotation
.pdf", "application/pdf", stream);
```

```

}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("UriAnnotation.pdf",
"application/pdf", stream);
}

```

Document Link Annotation

This annotation is used to navigate to a specific destination within the document.

[PdfDocumentLinkAnnotation](#) is used to add a document link annotation in PDF document.

C#

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a page
PdfPage page2 = document.Pages.Add();
//Creates a new rectangle
RectangleF docLinkAnnotationRectangle = new RectangleF(10, 40, 30, 30);
//Creates a new document link annotation.
PdfDocumentLinkAnnotation documentLinkAnnotation = new
PdfDocumentLinkAnnotation(docLinkAnnotationRectangle);
documentLinkAnnotation.AnnotationFlags = PdfAnnotationFlags.NoRotate;
documentLinkAnnotation.Text = "Document link annotation";
documentLinkAnnotation.Color = new PdfColor(Color.Navy);
//Sets the destination.
documentLinkAnnotation.Destination = new PdfDestination(page2);
documentLinkAnnotation.Destination.Location = new Point(10, 0);
documentLinkAnnotation.Destination.Zoom = 5;
//Adds this annotation to a new page.
page.Annotations.Add(documentLinkAnnotation);
//Saves the document to disk.
document.Save("DocumentLinkAnnotation.pdf");
document.Close(true);

```

VB.NET

```

'Creates a new PDF document
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Creates a page
Dim page2 As PdfPage = document.Pages.Add()
'Creates a new rectangle
Dim docLinkAnnotationRectangle As New RectangleF(10, 40, 30, 30)
'Creates a new document link annotation.
Dim documentLinkAnnotation As New
PdfDocumentLinkAnnotation(docLinkAnnotationRectangle)
documentLinkAnnotation.AnnotationFlags = PdfAnnotationFlags.NoRotate
documentLinkAnnotation.Text = "Document link annotation"
documentLinkAnnotation.Color = New PdfColor(Color.Navy)
'Sets the destination.
documentLinkAnnotation.Destination = New PdfDestination(page2)

```

```
documentLinkAnnotation.Destination.Location = New Point(10, 0)
documentLinkAnnotation.Destination.Zoom = 5
'Adds this annotation to a new page.
page.Annotations.Add(documentLinkAnnotation)
'Saves the document to disk.
document.Save("DocumentLinkAnnotation.pdf")
document.Close(True)
```

UWP

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a page
PdfPage page2 = document.Pages.Add();
//Creates a new rectangle
RectangleF docLinkAnnotationRectangle = new RectangleF(10, 40, 30, 30);
//Creates a new document link annotation.
PdfDocumentLinkAnnotation documentLinkAnnotation = new
PdfDocumentLinkAnnotation(docLinkAnnotationRectangle)
documentLinkAnnotation.AnnotationFlags = PdfAnnotationFlags.NoRotate;
documentLinkAnnotation.Text = "Document link annotation";
documentLinkAnnotation.Color = new PdfColor(Color.FromArgb(0, 0, 0, 128));
//Sets the destination.
documentLinkAnnotation.Destination = new PdfDestination(page2);
documentLinkAnnotation.Destination.Location = new PointF(10, 0);
documentLinkAnnotation.Destination.Zoom = 5;
//Adds this annotation to a new page.
page.Annotations.Add(documentLinkAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "DocumentLinkAnnotation.pdf");
```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a page
PdfPage page2 = document.Pages.Add();
//Creates a new rectangle
RectangleF docLinkAnnotationRectangle = new RectangleF(10, 40, 30, 30);
//Creates a new document link annotation.
PdfDocumentLinkAnnotation documentLinkAnnotation = new
PdfDocumentLinkAnnotation(docLinkAnnotationRectangle);
documentLinkAnnotation.AnnotationFlags = PdfAnnotationFlags.NoRotate;
documentLinkAnnotation.Text = "Document link annotation";
documentLinkAnnotation.Color = new PdfColor(Color.Navy);
//Sets the destination.
```

```

documentLinkAnnotation.Destination = new PdfDestination(page2);
documentLinkAnnotation.Destination.Location = new PointF(10, 0);
documentLinkAnnotation.Destination.Zoom = 5;
//Adds this annotation to a new page.
page.Annotations.Add(documentLinkAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "DocumentLinkAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a page
PdfPage page2 = document.Pages.Add();
//Creates a new rectangle
RectangleF docLinkAnnotationRectangle = new RectangleF(10, 40, 30, 30);
//Creates a new document link annotation.
PdfDocumentLinkAnnotation documentLinkAnnotation = new
PdfDocumentLinkAnnotation(docLinkAnnotationRectangle);
documentLinkAnnotation.AnnotationFlags = PdfAnnotationFlags.NoRotate;
documentLinkAnnotation.Text = "Document link annotation";
documentLinkAnnotation.Color = new PdfColor(Syncfusion.Drawing.Color.Navy);
//Sets the destination.
documentLinkAnnotation.Destination = new PdfDestination(page2);
documentLinkAnnotation.Destination.Location = new
Syncfusion.Drawing.PointF(10, 0);
documentLinkAnnotation.Destination.Zoom = 5;
//Adds this annotation to a new page.
page.Annotations.Add(documentLinkAnnotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DocumentLinkA
nnotation.pdf", "application/pdf", stream);
}

```

```

}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("DocumentLinkAnnotation.pdf", "application/pdf", stream);
}

```

Measurement Annotations

Essential PDF supports interactive measurement annotations, which measures the distance, area, and angle of the line segments.

The following measurement annotation types are supported in Essential PDF:

Line measurement annotation

The line measurement annotation is displayed as the straight line in the page. The distance of the line is measured automatically when you change the position of the line and is displayed in the pop-up window.

[PdfLineMeasurementAnnotation](#) to add a line measurement annotation to the page.

C#

```

//Creates a new PDF document.
PdfDocument document= new PdfDocument();
//Creates a new page.
PdfPage page = document.Pages.Add();
PdfFont font= new PdfStandardFont(PdfFontFamily.Helvetica, 10f, PdfFontStyle.Regular);
//Specifies the line end points.
int[] points = new int[] { 100, 750, 500, 750 };
//Creates the line measurement annotation
PdfLineMeasurementAnnotation lineMeasureAnnotation = new PdfLineMeasurementAnnotation(points);
//Assign author to the line measurement annotation.
lineMeasureAnnotation.Author = "Syncfusion";
//Assign subject to the line measurement annotation
lineMeasureAnnotation.Subject = "LineAnnotation";
//Assign unit to the line measurement annotation
lineMeasureAnnotation.Unit = PdfMeasurementUnit.Inch;
//Assign borderWidth to the line measurement annotation.
lineMeasureAnnotation.LineBorder.BorderWidth = 2;
//Assign font to the line measurement annotation.
lineMeasureAnnotation.Font = font;
//Assign color to the line measurement annotation.
lineMeasureAnnotation.Color = new PdfColor(Color.Red);
//Adds the line measurement annotation to a new page.
page.Annotations.Add(lineMeasureAnnotation);
//Saves the document to disk.
document.Save("LineMeasurementAnnotation.pdf");
//Close the document.
document.Close(true);

```

VB.NET

```

//Creates a new PDF document
Dim document As PdfDocument = New PdfDocument

```

```
//Creates a new page
Dim page As PdfPage = document.Pages.Add
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 10.0!,
PdfFontStyle.Regular)
//Specifies the line end points.
Dim points() As Integer = New Integer() {100, 750, 500, 750}
//Creates the line measurement annotation
Dim lineMeasureAnnotation As PdfLineMeasurementAnnotation = New
PdfLineMeasurementAnnotation(points)
'Assign author to the line measurement annotation
lineMeasureAnnotation.Author = "Syncfusion"
'Assign subject to the line measurement annotation
lineMeasureAnnotation.Subject = "LineAnnotation"
'Assign unit to the line measurement annotation
lineMeasureAnnotation.Unit = PdfMeasurementUnit.Inch
'Assign borderWidth to the line measurement annotation
lineMeasureAnnotation.lineBorder.BorderWidth = 2
'Assign font to the line measurement annotation
lineMeasureAnnotation.Font = font
'Assign color to the line measurement annotation
lineMeasureAnnotation.Color = New PdfColor(Color.Red)
'Adds the line measurement annotation to a new page
page.Annotations.Add(lineMeasureAnnotation)
'Saves the document to disk
document.Save("LineMeasurementAnnotation.pdf")
'Close the document
document.Close(True)
```

UWP

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 10f,
PdfFontStyle.Regular);
//Specifies the line end points
int[] points = new int[] { 100, 750, 500, 750 };
//Creates the line measurement annotation
PdfLineMeasurementAnnotation lineMeasureAnnotation = new
PdfLineMeasurementAnnotation(points);
//Assign author to the line measurement annotation
lineMeasureAnnotation.Author = "Syncfusion";
//Assign subject to the line measurement annotation
lineMeasureAnnotation.Subject = "LineAnnotation";
//Assign unit to the line measurement annotation
lineMeasureAnnotation.Unit = PdfMeasurementUnit.Inch;
//Assign borderWidth to the line measurement annotation
lineMeasureAnnotation.lineBorder.BorderWidth = 2;
//Assign font to the line measurement annotation
lineMeasureAnnotation.Font = font;
//Assign color to the line measurement annotation
lineMeasureAnnotation.Color = new PdfColor(Color.FromArgb(255, 255, 0, 0));
//Adds the line measurement annotation to a new page
page.Annotations.Add(lineMeasureAnnotation);
MemoryStream stream = new MemoryStream();
```

```
//Save the PDF document to stream
document.Save(stream);
//Close the document
document.Close(true);
```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 10f,
PdfFontStyle.Regular);
//Specifies the line end points
int[] points = new int[] { 100, 750, 500, 750 };
//Creates the line measurement annotation
PdfLineMeasurementAnnotation lineMeasureAnnotation = new
PdfLineMeasurementAnnotation(points);
//Assign author to the line measurement annotation
lineMeasureAnnotation.Author = "Syncfusion";
//Assign subject to the line measurement annotation
lineMeasureAnnotation.Subject = "LineAnnotation";
//Assign unit to the line measurement annotation
lineMeasureAnnotation.Unit = PdfMeasurementUnit.Inch;
//Assign borderWidth to the line measurement annotation
lineMeasureAnnotation.LineBorder.BorderWidth = 2;
//Assign font to the line measurement annotation
lineMeasureAnnotation.Font = font;
//Assign color to the line measurement annotation
lineMeasureAnnotation.Color = new PdfColor(Syncfusion.Drawing.Color.Red);
//Adds the line measurement annotation to a new page
page.Annotations.Add(lineMeasureAnnotation);
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
document.Save(stream);
stream.Position=0;
//Close the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "LineMeasurementAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 10f,
PdfFontStyle.Regular);
//Specifies the line end points
int[] points = new int[] { 100, 750, 500, 750 };
```



```

//Creates the line measurement annotation
PdfLineMeasurementAnnotation lineMeasureAnnotation = new
PdfLineMeasurementAnnotation(points);
//Assign author to the line measurement annotation
lineMeasureAnnotation.Author = "Syncfusion";
//Assign subject to the line measurement annotation
lineMeasureAnnotation.Subject = "LineAnnotation";
//Assign unit to the line measurement annotation
lineMeasureAnnotation.Unit = PdfMeasurementUnit.Inch;
//Assign borderWidth to the line measurement annotation
lineMeasureAnnotation.lineBorder.BorderWidth = 2;
//Assign font to the line measurement annotation
lineMeasureAnnotation.Font = font;
//Assign color to the line measurement annotation
lineMeasureAnnotation.Color = new PdfColor(Syncfusion.Drawing.Color.Red);
//Adds the line measurement annotation to a new page
page.Annotations.Add(lineMeasureAnnotation);
MemoryStream stream = new MemoryStream();
//Save the document into stream.
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("LineMeasureme
ntAnnotation.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("LineMeasurementAnnotation
.pdf", "application/pdf", stream);
}

```

Square measurement annotation

The square measurement annotation is displayed as square shape in the page. The area of the square is measured when you change the square bound and is displayed in the pop-up window.

[PdfSquareMeasurementAnnotation](#) is used to add a square measurement annotation to the page.

C#

```

//Creates a new PDF document
PdfDocument document= new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
RectangleF rect = new RectangleF(10, 100, 100,100);
//Creates the square measurement annotation
PdfSquareMeasurementAnnotation squareMeasureAnnotation = new
PdfSquareMeasurementAnnotation(rect);
//Assign author to the square measurement annotation
squareMeasureAnnotation.Author = "Syncfusion";

```

```
//Assign subject to the square measurement annotation
squareMeasureAnnotation.Subject = "Square measurement annotation";
//Assign color to the square measurement annotation
squareMeasureAnnotation.Color = new PdfColor(Color.Red);
//Assign measurement unit to the square measurement annotation
squareMeasureAnnotation.Unit = PdfMeasurementUnit.Centimeter;
//Adds the square measurement annotation to a page
page.Annotations.Add(squareMeasureAnnotation);
//Saves the document to disk
document.Save("SquareMeasurementAnnotation.pdf");
//Close the document
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
Dim rect As New RectangleF(10, 100, 100, 100)
'Creates the square measurement annotation
Dim squareMeasureAnnotation As New PdfSquareMeasurementAnnotation(rect)
'Assign author to the square measurement annotation
squareMeasureAnnotation.Author = "Syncfusion"
'Assign subject to the square measurement annotation
squareMeasureAnnotation.Subject = "Square measurement annotation"
'Assign color to the square measurement annotation
squareMeasureAnnotation.Color = New PdfColor(Color.Red)
'Assign measurement unit to the square measurement annotation
squareMeasureAnnotation.Unit = PdfMeasurementUnit.Centimeter
'Adds the square measurement annotation to a page
page.Annotations.Add(squareMeasureAnnotation)
'Saves the document to disk
document.Save("SquareMeasurementAnnotation.pdf")
'Close the document
document.Close(True)
```

UWP

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
RectangleF rect = new RectangleF(10, 100, 100, 100);
//Creates the square measurement annotation
PdfSquareMeasurementAnnotation squareMeasureAnnotation = new
PdfSquareMeasurementAnnotation(rect);
//Assign author to the square measurement annotation
squareMeasureAnnotation.Author = "Syncfusion";
//Assign subject to the square measurement annotation
squareMeasureAnnotation.Subject = "Square measurement annotation";
//Assign color to the square measurement annotation
squareMeasureAnnotation.Color = new PdfColor(Color.FromArgb(255, 255, 0,
0));
//Assign measurement unit to the square measurement annotation
squareMeasureAnnotation.Unit = PdfMeasurementUnit.Centimeter;
```

```
//Adds the square measurement annotation to a page
page.Annotations.Add(squareMeasureAnnotation);
MemoryStream stream = new MemoryStream();
//Save the document into stream.
document.Save(stream);
//Close the document
document.Close(true);
```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
Syncfusion.Drawing.RectangleF rect = new Syncfusion.Drawing.RectangleF(10,
100, 100, 100);
//Creates the square measurement annotation
PdfSquareMeasurementAnnotation squareMeasureAnnotation = new
PdfSquareMeasurementAnnotation(rect);
//Assign author to the square measurement annotation
squareMeasureAnnotation.Author = "Syncfusion";
//Assign subject to the square measurement annotation
squareMeasureAnnotation.Subject = "Square measurement annotation";
//Assign color to the square measurement annotation
squareMeasureAnnotation.Color = new PdfColor(Syncfusion.Drawing.Color.Red);
//Assign measurement unit to the square measurement annotation
squareMeasureAnnotation.Unit = PdfMeasurementUnit.Centimeter;
//Adds the square measurement annotation to a page
page.Annotations.Add(squareMeasureAnnotation);
MemoryStream stream = new MemoryStream();
//Save the document into stream
document.Save(stream);
stream.Position=0;
//Close the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "SquareMeasurementAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
Syncfusion.Drawing.RectangleF rect = new Syncfusion.Drawing.RectangleF(10,
100, 100, 100);
//Creates the square measurement annotation
PdfSquareMeasurementAnnotation squareMeasureAnnotation = new
PdfSquareMeasurementAnnotation(rect);
//Assign author to the square measurement annotation
squareMeasureAnnotation.Author = "Syncfusion";
```

```

//Assign subject to the square measurement annotation
squareMeasureAnnotation.Subject = "Square measurement annotation";
//Assign color to the square measurement annotation
squareMeasureAnnotation.Color = new PdfColor(Syncfusion.Drawing.Color.Red);
//Assign measurement unit to the square measurement annotation
squareMeasureAnnotation.Unit = PdfMeasurementUnit.Centimeter;
//Adds the square measurement annotation to a page
page.Annotations.Add(squareMeasureAnnotation);
MemoryStream stream = new MemoryStream();
//Saves the document to disk
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("SquareMeasure
mentAnnotation.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("SquareMeasurementAnnotati
on.pdf", "application/pdf", stream);
}

```

Circle measurement annotation

The circle measurement annotation is displayed as circle shape in the page. The radius or diameter distance of the circle is measured and the value is displayed in the pop-up window.

[PdfCircleMeasurementAnnotation](#) is used to add a circle measurement annotation to the page.

C#

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
RectangleF rect = new RectangleF(10, 100, 100, 100);
//Creates the circle measurement annotation
PdfCircleMeasurementAnnotation circleMeasureAnnotation = new
PdfCircleMeasurementAnnotation(rect);
//Assign author to the circle measurement annotation
circleMeasureAnnotation.Author = "Syncfusion";
//Assign subject to the circle measurement annotation
circleMeasureAnnotation.Subject = "Circle measurement annotation";
//Assign color to the square measurement annotation
circleMeasureAnnotation.Color = new PdfColor(Color.Red);
//Assign measurement unit to the circle measurement annotation
circleMeasureAnnotation.Unit = PdfMeasurementUnit.Centimeter;
//Sets the measurementType to the circle measurement annotation
circleMeasureAnnotation.MeasurementType = PdfCircleMeasurementType.Diameter;
//Adds the circle measurement annotation to a page

```

```

page.Annotations.Add(circleMeasureAnnotation);
//Saves the document to disk
document.Save("CircleMeasurementAnnotation.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Creates a new PDF document
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
Dim rect As New RectangleF(10, 100, 100, 100)
'Creates the circle measurement annotation
Dim circleMeasureAnnotation As New PdfCircleMeasurementAnnotation(rect)
'Assign author to the circle measurement annotation
circleMeasureAnnotation.Author = "Syncfusion"
'Assign subject to the circle measurement annotation
circleMeasureAnnotation.Subject = "Circle measurement annotation"
'Assign color to the square measurement annotation
circleMeasureAnnotation.Color = New PdfColor(Color.Red)
'Assign measurement unit to the circle measurement annotation
circleMeasureAnnotation.Unit = PdfMeasurementUnit.Centimeter
'Sets the measurementType to the circle measurement annotation
circleMeasureAnnotation.MeasurementType = PdfCircleMeasurementType.Diameter
'Adds the circle measurement annotation to a page
page.Annotations.Add(circleMeasureAnnotation)
'Saves the document to disk
document.Save("CircleMeasurementAnnotation.pdf")
'Close the document
document.Close(True)

```

UWP

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
RectangleF rect = new RectangleF(10, 100, 100, 100);
//Creates the circle measurement annotation
PdfCircleMeasurementAnnotation circleMeasureAnnotation = new
PdfCircleMeasurementAnnotation(rect);
//Assign author to the circle measurement annotation
circleMeasureAnnotation.Author = "Syncfusion";
//Assign subject to the circle measurement annotation
circleMeasureAnnotation.Subject = "Circle measurement annotation";
//Assign color to the square measurement annotation
circleMeasureAnnotation.Color = new PdfColor(Color.FromArgb(255, 255, 0,
0));
//Assign measurement unit to the circle measurement annotation
circleMeasureAnnotation.Unit = PdfMeasurementUnit.Centimeter;
//Sets the measurementType to the circle measurement annotation
circleMeasureAnnotation.MeasurementType = PdfCircleMeasurementType.Diameter;
//Adds the circle measurement annotation to a page
page.Annotations.Add(circleMeasureAnnotation);
//Save the document into stream.

```

```
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
Syncfusion.Drawing.RectangleF rect = new Syncfusion.Drawing.RectangleF(10,
100, 100, 100);
//Creates the circle measurement annotation
PdfCircleMeasurementAnnotation circleMeasureAnnotation = new
PdfCircleMeasurementAnnotation(rect);
//Assign author to the circle measurement annotation
circleMeasureAnnotation.Author = "Syncfusion";
//Assign subject to the circle measurement annotation
circleMeasureAnnotation.Subject = "Circle measurement annotation";
//Assign color to the square measurement annotation
circleMeasureAnnotation.Color = new PdfColor(Syncfusion.Drawing.Color.Red);
//Assign measurement unit to the circle measurement annotation
circleMeasureAnnotation.Unit = PdfMeasurementUnit.Centimeter;
//Sets the measurementType to the circle measurement annotation
circleMeasureAnnotation.MeasurementType = PdfCircleMeasurementType.Diameter;
//Adds the circle measurement annotation to a page
page.Annotations.Add(circleMeasureAnnotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "CircleMeasurementAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
Syncfusion.Drawing.RectangleF rect = new Syncfusion.Drawing.RectangleF(10,
100, 100, 100);
//Creates the circle measurement annotation
PdfCircleMeasurementAnnotation circleMeasureAnnotation = new
PdfCircleMeasurementAnnotation(rect);
//Assign author to the circle measurement annotation
circleMeasureAnnotation.Author = "Syncfusion";
//Assign subject to the circle measurement annotation
```

```

circleMeasureAnnotation.Subject = "Circle measurement annotation";
//Assign color to the square measurement annotation
circleMeasureAnnotation.Color = new PdfColor(Syncfusion.Drawing.Color.Red);
//Assign measurement unit to the circle measurement annotation
circleMeasureAnnotation.Unit = PdfMeasurementUnit.Centimeter;
//Sets the measurementType to the circle measurement annotation
circleMeasureAnnotation.MeasurementType = PdfCircleMeasurementType.Diameter;
//Adds the circle measurement annotation to a page
page.Annotations.Add(circleMeasureAnnotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("CircleMeasure
mentAnnotation.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("CircleMeasurementAnnotati
on.pdf", "application/pdf", stream);
}

```

Angle measurement annotation

The angle measurement annotation calculates the angle between three points and draws arc between three points. The angle of the annotation is displayed in the pop-up window.

[PdfAngleMeasurementAnnotation](#) helps you to add angle measurement annotation to the page.

C#

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
PointF[] points = new PointF[] { new PointF(100, 700), new PointF(150, 650),
new PointF(100, 600) };
//Creates the angel measurement annotation
PdfAngleMeasurementAnnotation angleMeasureAnnotation = new
PdfAngleMeasurementAnnotation(points);
//Set font to the angle measurement annotation
angleMeasureAnnotation.Font = new PdfStandardFont(PdfFontFamily.Helvetica,
12f, PdfFontStyle.Regular);
//Assign color to the angle measurement annotation
angleMeasureAnnotation.Color = Color.Red;
//Adds angle measurement annotation
page.Annotations.Add(angleMeasureAnnotation);
//Saves the document to disk

```

```
document.Save("AngleMeasurementAnnotation.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
Dim points As PointF() = New PointF() {New PointF(100, 700), New PointF(150, 650), New PointF(100, 600)}
'Creates the angel measurement annotation
Dim angleMeasureAnnotation As New PdfAngleMeasurementAnnotation(points)
'Set font to the angle measurement annotation
angleMeasureAnnotation.Font = New PdfStandardFont(PdfFontFamily.Helvetica, 12.0F, PdfFontStyle.Regular)
'Assign color to the angle measurement annotation
angleMeasureAnnotation.Color = Color.Red
'Adds angle measurement annotation
page.Annotations.Add(angleMeasureAnnotation)
'Saves the document to disk
document.Save("AngleMeasurementAnnotation.pdf")
document.Close(True)
```

UWP

```
//PDF supports angle measurement annotation only in Windows Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Web.
```

ASP.NET CORE

```
//PDF supports angle measurement annotation only in Windows Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Web.
```

XAMARIN

```
//PDF supports angle measurement annotation only in Windows Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Web.
```

Modifying the annotations

Essential PDF allows you to modify the annotation of existing document. The following code illustrates this.

C#

```
//Loads the document
PdfLoadedDocument lDoc = new PdfLoadedDocument("inputAnnotation.pdf");
//Gets the first page from the document
PdfLoadedPage page = lDoc.Pages[0] as PdfLoadedPage;
//Gets the annotation collection
PdfLoadedAnnotationCollection annotations = page.Annotations;
//Gets the first annotation and modify the properties
PdfLoadedPopupAnnotation popUp = annotations[0] as PdfLoadedPopupAnnotation;
popUp.Border = new PdfAnnotationBorder(4, 0, 0);
```



```

popUp.Color = new PdfColor(Color.Red);
popUp.Text = "Modified annotation";
//Saves the document
lDoc.Save("sample.pdf");
lDoc.Close(true);

```

VB.NET

```

'Loads the document
Dim lDoc As New PdfLoadedDocument("inputAnnotation.pdf")
'Gets the first page from the document
Dim page As PdfLoadedPage = TryCast(lDoc.Pages(0), PdfLoadedPage)
'Gets the annotation collections
Dim annotations As PdfLoadedAnnotationCollection = page.Annotations
'Gets the annotation at index 0 and modifying the properties
Dim popUp As PdfLoadedPopupAnnotation = TryCast(annotations(0),
PdfLoadedPopupAnnotation)
popUp.Border = New PdfAnnotationBorder(4, 0, 0)
popUp.Color = New PdfColor(Color.Red)
popUp.Text = "Modified annotation"
'Saves the document
lDoc.Save("sample.pdf")
lDoc.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await lDoc.OpenAsync(file);
//Gets the first page from the document
PdfLoadedPage page = lDoc.Pages[0] as PdfLoadedPage;
//Gets the annotation collection
PdfLoadedAnnotationCollection annotations = page.Annotations;
//Gets the first annotation and modify the properties
PdfLoadedPopupAnnotation popUp = annotations[0] as PdfLoadedPopupAnnotation;
popUp.Border = new PdfAnnotationBorder(4, 0, 0);
popUp.Color = new PdfColor(Color.FromArgb(0,255,0,0));
popUp.Text = "Modified annotation";
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await lDoc.SaveAsync(stream);
//Close the document
lDoc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "sample.pdf");

```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("inputAnnotation.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Gets the first page from the document
PdfLoadedPage page = lDoc.Pages[0] as PdfLoadedPage;
//Gets the annotation collection
PdfLoadedAnnotationCollection annotations = page.Annotations;
//Gets the first annotation and modify the properties
PdfLoadedPopupAnnotation popUp = annotations[0] as PdfLoadedPopupAnnotation;
popUp.Border = new PdfAnnotationBorder(4, 0, 0);
popUp.Color = new PdfColor(Color.Red);
popUp.Text = "Modified annotation";
//Save the document into stream
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
stream.Position = 0;
//Closes the document
lDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
inputAnnotation.pdf");
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Gets the first page from the document
PdfLoadedPage page = lDoc.Pages[0] as PdfLoadedPage;
//Gets the annotation collection
PdfLoadedAnnotationCollection annotations = page.Annotations;
//Gets the first annotation and modify the properties
PdfLoadedPopupAnnotation popUp = annotations[0] as PdfLoadedPopupAnnotation;
popUp.Border = new PdfAnnotationBorder(4, 0, 0);
popUp.Color = new PdfColor(Syncfusion.Drawing.Color.Red);
popUp.Text = "Modified annotation";
//Save the document into stream.
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
//Close the document.
lDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}
```

Removing annotations from an existing PDF

You can remove the annotation from the annotation collection, represented by the [PdfLoadedAnnotationCollection](#) of the loaded page. The following code illustrates this.

C#

```
//Loads the document
PdfLoadedDocument lDoc = new PdfLoadedDocument("inputAnnotation.pdf");
//Gets the first page of the document
PdfLoadedPage page = lDoc.Pages[0] as PdfLoadedPage;
//Gets the annotation collection
PdfLoadedAnnotationCollection annotations = page.Annotations;
//Removes the first annotation
annotations.RemoveAt(0);
//Saves the document
lDoc.Save("sample.pdf");
lDoc.Close(true);
```

VB.NET

```
'Loads the document
Dim lDoc As New PdfLoadedDocument("inputAnnotation.pdf")
'Gets the first page from the document
Dim page As PdfLoadedPage = TryCast(lDoc.Pages(0), PdfLoadedPage)
'Gets the annotation collection
Dim annotations As PdfLoadedAnnotationCollection = page.Annotations
'Removes the first annotation
annotations.RemoveAt(0)
'Saves the document
lDoc.Save("sample.pdf")
lDoc.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await lDoc.OpenAsync(file);
//Gets the first page of the document
PdfLoadedPage page = lDoc.Pages[0] as PdfLoadedPage;
```

```
//Gets the annotation collection
PdfLoadedAnnotationCollection annotations = page.Annotations;
//Removes the first annotation
annotations.RemoveAt(0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await lDoc.SaveAsync(stream);
//Close the document
lDoc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "sample.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("inputAnnotation.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Gets the first page of the document
PdfLoadedPage page = lDoc.Pages[0] as PdfLoadedPage;
//Gets the annotation collection
PdfLoadedAnnotationCollection annotations = page.Annotations;
//Removes the first annotation
annotations.RemoveAt(0);
//Save the document into stream
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
stream.Position = 0;
//Closes the document
lDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
inputAnnotation.pdf");
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Gets the first page of the document
PdfLoadedPage page = lDoc.Pages[0] as PdfLoadedPage;
//Gets the annotation collection
PdfLoadedAnnotationCollection annotations = page.Annotations;
//Removes the first annotation
annotations.RemoveAt(0);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
//Close the document.
```

```

lDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

Importing annotations from FDF file

FDF stands for Forms Data Format. FDF is a file format for representing annotations present in a PDF document. You can import annotation data from the FDF file to PDF using the [ImportAnnotations](#) method in [PdfLoadedDocument](#) class.

C#

```

//Loads the document
PdfLoadedDocument lDoc = new PdfLoadedDocument("input.pdf");
//Import annotation data from FDF file
lDoc.ImportAnnotations("Annotations.fdf", AnnotationDataFormat.Fdf);
//Saves the document
lDoc.Save("Annotation.pdf");
lDoc.Close(true);

```

VB.NET

```

'Loads the document
Dim lDoc As New PdfLoadedDocument("input.pdf")
'Import annotation data from FDF file
lDoc.ImportAnnotations("Annotations.fdf", AnnotationDataFormat.Fdf)
'Saves the document
lDoc.Save("Annotation.pdf")
lDoc.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through the Open method of
PdfLoadedDocument class
await lDoc.OpenAsync(file);
//Load the FDF file stream from the disk

```

```

Stream fdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Annotations.fdf");
'Import annotation data from FDF stream
lDoc.ImportAnnotations(fdfStream, AnnotationDataFormat.Fdf)
MemoryStream stream = new MemoryStream();
await lDoc.SaveAsync(stream);
//Close the document
lDoc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Annotation.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Import annotation data from FDF stream
FileStream fdfStream = new FileStream("Annotations.fdf", FileMode.Open,
FileAccess.Read);
lDoc.ImportAnnotations(fdfStream, AnnotationDataFormat.Fdf)
//Save the document into stream
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
stream.Position = 0;
//Closes the document
lDoc.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Annotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Import annotation data from FDF stream
Stream fdfStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Annotations.fdf");
lDoc.ImportAnnotations(fdfStream, AnnotationDataFormat.Fdf)
//Save the document into stream
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
//Close the document
lDoc.Close(true);
//Save the stream into PDF file

```

```
//The operation in Save under Xamarin varies between Windows Phone, Android, and iOS platforms. Refer to the PDF/Xamarin section for respective code samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Annotation.pdf", "application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Annotation.pdf", "application/pdf", stream);
}
```

Importing annotations from XFDF file

XFDF stands for XML Forms Data Format. XFDF is the XML version of FDF for representing annotations that are contained in a PDF document. You can import annotation data from the XFDF file to PDF using the [ImportAnnotations](#) method in [PdfLoadedDocument](#) class.

C#

```
//Loads the document
PdfLoadedDocument lDoc = new PdfLoadedDocument("input.pdf");
//Import annotation data from XFDF file
lDoc.ImportAnnotations("Annotations.xfdf", AnnotationDataFormat.XFdf);
//Saves the document
lDoc.Save("Annotation.pdf");
lDoc.Close(true);
```

VB.NET

```
'Loads the document
Dim lDoc As New PdfLoadedDocument("input.pdf")
'Import annotation data from XFDF file
lDoc.ImportAnnotations("Annotations.xfdf", AnnotationDataFormat.XFdf)
'Saves the document
lDoc.Save("Annotation.pdf")
lDoc.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through the Open method of PdfLoadedDocument class
await lDoc.OpenAsync(file);
//Load the XFDF file stream from the disk
```

```

Stream xfdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Annotations.xfdf");
'Import annotation data from XFDF stream
lDoc.ImportAnnotations(xfdfStream, AnnotationDataFormat.XFdf)
MemoryStream stream = new MemoryStream();
await lDoc.SaveChangesAsync();
//Close the document
lDoc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Annotation.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Import annotation data from XFDF stream
FileStream xfdfStream = new FileStream("Annotations.xfdf", FileMode.Open,
FileAccess.Read);
lDoc.ImportAnnotations(xfdfStream, AnnotationDataFormat.XFdf)
//Save the document into stream
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
stream.Position = 0;
//Closes the document
lDoc.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Annotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Import annotation data from XFDF stream
Stream xfdfStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Annotations.xfdf");
lDoc.ImportAnnotations(xfdfStream, AnnotationDataFormat.XFdf)
//Save the document into stream
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
//Close the document
lDoc.Close(true);
//Save the stream into PDF file

```



```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Annotation.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Annotation.pdf",
"application/pdf", stream);
}
```

Exporting annotations to FDF file

To export annotation data to the FDF file from PDF document, you can use the [ExportAnnotations](#) method in [PdfLoadedDocument](#) class.

C#

```
//Loads the document
PdfLoadedDocument lDoc = new PdfLoadedDocument("input.pdf");
//Export annotation data to FDF file
lDoc.ExportAnnotations("Annotations.fdf", AnnotationDataFormat.Fdf);
//Close the document
lDoc.Close(true);
```

VB.NET

```
'Loads the document
Dim lDoc As New PdfLoadedDocument("input.pdf")
'Export annotation data to FDF file
lDoc.ExportAnnotations("Annotations.fdf", AnnotationDataFormat.Fdf)
'Close the document
lDoc.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through the Open method of
PdfLoadedDocument class
await lDoc.OpenAsync(file);
//Load the FDF file stream from the disk
Stream fdfStream = new MemoryStream();
//Export annotation data from FDF stream
lDoc.ExportAnnotations(fdfStream, AnnotationDataFormat.Fdf)
//Save the fdfStream as FDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
```

```
Save(fdfStream, "Annotations.fdf");
//Close the document
lDoc.Close(true);
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Export annotation data from FDF stream
Stream fdfStream = new MemoryStream();
lDoc.ExportAnnotations(fdfStream, AnnotationDataFormat.Fdf)
//Close the document
lDoc.Close(true);
fdfStream.Position = 0;
//Defining the ContentType for FDF file
string contentType = "application/fdf";
//Define the file name
string fileName = "Annotations.fdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(fdfStream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Export annotation data from FDF stream
Stream fdfStream = new MemoryStream();
lDoc.ExportAnnotations(fdfStream, AnnotationDataFormat.Fdf)
//Close the document
lDoc.Close(true);
//Save the stream into FDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Annotations.f
df", "application/fdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Annotations.fdf",
"application/fdf", fdfStream);
}
```

Exporting annotations to XFDF file

To export annotation data to the XFDF file from PDF document, you can use the [ExportAnnotations](#) method in [PdfLoadedDocument](#) class.

C#

```
//Loads the document
PdfLoadedDocument lDoc = new PdfLoadedDocument("input.pdf");
//Export annotation data to XFDF file
lDoc.ExportAnnotations("Annotations.xfdf", AnnotationDataFormat.XFdf);
//Close the document
lDoc.Close(true);
```

VB.NET

```
'Loads the document
Dim lDoc As New PdfLoadedDocument("input.pdf")
'Export annotation data to XFDF file
lDoc.ExportAnnotations("Annotations.xfdf", AnnotationDataFormat.XFdf)
'Close the document
lDoc.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through the Open method of
PdfLoadedDocument class
await lDoc.OpenAsync(file);
//Load the XFDF file stream from the disk
Stream xfdfStream = new MemoryStream();
//Export annotation data from XFDF stream
lDoc.ExportAnnotations(xfdfStream, AnnotationDataFormat.XFdf)
//Save the xfdfStream as XFDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(xfdfStream, "Annotations.xfdf");
//Close the document
lDoc.Close(true);
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Export annotation data from XFDF stream
Stream xfdfStream = new MemoryStream();
lDoc.ExportAnnotations(xfdfStream, AnnotationDataFormat.XFdf)
//Close the document
lDoc.Close(true);
```

```
xfdfStream.Position = 0;
//Defining the ContentType for XFDF file
string contentType = "application/xfdf";
//Define the file name
string fileName = "Annotations.xfdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(xfdfStream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Export annotation data from XFDF stream
Stream xfdfStream = new MemoryStream();
lDoc.ExportAnnotations(xfdfStream, AnnotationDataFormat.XFdf)
//Close the document
lDoc.Close(true);
//Save the stream into XFDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Annotations.x
fdf", "application/xfdf", fdfStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Annotations.xfdf",
"application/xfdf", fdfStream);
}
```

Adding comments and review status to the PDF annotation

The PDF annotations may have an author-specific state associated with them. The state is not specified in the annotation itself, but it represents a separate text annotation ([Popup Annotation](#)).

The Essential PDF supports adding the annotation comments and review status to the PDF document annotations.

Adding comments to the PDF annotation

The following code example explains how to add comments to the PDF annotation.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create a new page
PdfPage page = document.Pages.Add();
//Create new rectangle annotation
PdfRectangleAnnotation rectangleAnnotation = new PdfRectangleAnnotation(new
RectangleF(0, 0, 100, 50), "Rectangle Annotation");
```

```

//Set author
rectangleAnnotation.Author = "Syncfusion";
rectangleAnnotation.Border.BorderWidth = 1;
rectangleAnnotation.Color = Color.Red;
rectangleAnnotation.ModifiedDate = DateTime.Now;
//Create a new comment annotation
PdfPopupAnnotation comment = new PdfPopupAnnotation();
//Set author
comment.Author = "John";
//Set comment text
comment.Text = "This is first comment";
//Set modification date
comment.ModifiedDate = DateTime.Now;
//Set subject
comment.Subject = "Annotation Comments";
//Add the comments to the annotation
rectangleAnnotation.Comments.Add(comment);
//Add the annotation to the PDF page
page.Annotations.Add(rectangleAnnotation);
//Save the document to disk
document.Save("Output.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Create a new page
Dim page As PdfPage = document.Pages.Add
'Create new rectangle annotation
Dim rectangleAnnotation As PdfRectangleAnnotation = New
PdfRectangleAnnotation(New RectangleF(0, 0, 100, 50), "Rectangle
Annotation")
'Set author
rectangleAnnotation.Author = "Syncfusion"
rectangleAnnotation.Border.BorderWidth = 1
rectangleAnnotation.Color = Color.Red
rectangleAnnotation.ModifiedDate = DateTime.Now
Dim comment As PdfPopupAnnotation = New PdfPopupAnnotation
'Set author
comment.Author = "John"
'Set Text
comment.Text = "This is first comment"
'Set modification date.
comment.ModifiedDate = DateTime.Now
'Set subject
comment.Subject = "Annotation Comments"
'Add the comment to the annotation.
rectangleAnnotation.Comments.Add(comment)
'Add the annotation to the PDF page.
page.Annotations.Add(rectangleAnnotation)
'Save the document to disk.
document.Save("Output.pdf")
'Close the document
document.Close(true)

```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create a new page
PdfPage page = document.Pages.Add();
//Create new rectangle annotation
PdfRectangleAnnotation rectangleAnnotation = new PdfRectangleAnnotation(new
RectangleF(0, 0, 100, 50), "Rectangle Annotation");
//Set author
rectangleAnnotation.Author = "Syncfusion";
rectangleAnnotation.Border.BorderWidth = 1;
rectangleAnnotation.Color = Color.Red;
rectangleAnnotation.ModifiedDate = DateTime.Now;
//Create a new comment annotation
PdfPopupAnnotation comment = new PdfPopupAnnotation();
//Set author
comment.Author = "John";
//Set Text
comment.Text = "This is first comment";
//Set modification date
comment.ModifiedDate = DateTime.Now;
//Set subject
comment.Subject = "Annotation Comments";
//Add the comment to the annotation
rectangleAnnotation.Comments.Add(comment);
//Add the annotation to the PDF page
page.Annotations.Add(rectangleAnnotation);
//Saves the document
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create a new page
PdfPage page = document.Pages.Add();
//Create new rectangle annotation
PdfRectangleAnnotation rectangleAnnotation = new PdfRectangleAnnotation(new
RectangleF(0, 0, 100, 50), "Rectangle Annotation");
//Set author
rectangleAnnotation.Author = "Syncfusion";
rectangleAnnotation.Border.BorderWidth = 1;
rectangleAnnotation.Color = Color.Red;
rectangleAnnotation.ModifiedDate = DateTime.Now;
//Create a new comment annotation
PdfPopupAnnotation comment = new PdfPopupAnnotation();
//Set author
```

```

comment.Author = "John";
//Set Text
comment.Text = "This is first comment";
//Set modification date
comment.ModifiedDate = DateTime.Now;
//Set subject
comment.Subject = "Annotation Comments";
//Add the comment to the annotation
rectangleAnnotation.Comments.Add(comment);
//Add the annotation to the PDF page
page.Annotations.Add(rectangleAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create a new page
PdfPage page = document.Pages.Add();
//Create new rectangle annotation
PdfRectangleAnnotation rectangleAnnotation = new PdfRectangleAnnotation(new
RectangleF(0, 0, 100, 50), "Rectangle Annotation");
//Set author
rectangleAnnotation.Author = "Syncfusion";
rectangleAnnotation.Border.BorderWidth = 1;
rectangleAnnotation.Color = Color.Red;
rectangleAnnotation.ModifiedDate = DateTime.Now;
//Create a new comments annotation
PdfPopupAnnotation comment = new PdfPopupAnnotation();
//Set author
comment.Author = "John";
//Set Text
comment.Text = "This is first comment";
//Set modification date
comment.ModifiedDate = DateTime.Now;
//Set subject
comment.Subject = "Annotation Comments";
//Add the comment to the annotation
rectangleAnnotation.Comments.Add(comment);
//Add the annotation to the PDF page
page.Annotations.Add(rectangleAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);

```

```

//Close the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following code example explains how to add comments to the existing PDF annotation.

C#

```

//Load the PDF document
PdfLoadedDocument ldoc = new PdfLoadedDocument("Input.pdf");
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing PDF annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Create a new comment annotation
PdfPopupAnnotation comment = new PdfPopupAnnotation();
//Set author
comment.Author = "John";
//Set Text
comment.Text = "This is first comment";
//Set modification date
comment.ModifiedDate = DateTime.Now;
//Set subject
comment.Subject = "Annotation Comments";
//Add the comment to the annotation
loadedRectangleAnnotation.Comments.Add(comment);
//Save the document
ldoc.Save("Output.pdf");
//Close the document
ldoc.Close(true);

```

VB.NET

```

'Load the PDF document
Dim ldoc As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the existing PDF page
Dim lpage As PdfLoadedPage = CType(ldoc.Pages(0), PdfLoadedPage)
'Get the existing annotations
Dim annots As PdfLoadedAnnotationCollection = lpage.Annotations

```



```

'Get the existing rectangle annotation
Dim loadedRectangleAnnotation As PdfLoadedRectangleAnnotation =
CType(annots(0), PdfLoadedRectangleAnnotation)
'Get the existing rectangle annotation
Dim comment As PdfPopupAnnotation = New PdfPopupAnnotation
'Set author
comment.Author = "John"
'Set Text
comment.Text = "This is first comment"
'Set modification date
comment.ModifiedDate = DateTime.Now
'Set subject
comment.Subject = "Annotation Comments"
'Add the comment to the annotation.
loadedRectangleAnnotation.Comments.Add(comment)
'Save the document
ldoc.Save("Output.pdf")
'Close the document
ldoc.Close(true)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await lDoc.OpenAsync(file);
//Get the existing PDF page
PdfLoadedPage lpage = lDoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Create a new comment annotation
PdfPopupAnnotation comment = new PdfPopupAnnotation();
//Set author
comment.Author = "John";
//Set Text
comment.Text = "This is first comment";
//Set modification date
comment.ModifiedDate = DateTime.Now;
//Set subject
comment.Subject = "Annotation Comments";
//Add the comments to the annotation
loadedRectangleAnnotation.Comments.Add(comment);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await lDoc.SaveAsync(stream);
//Close the document
lDoc.Close(true);

```

```
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("inputAnnotation.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage lpage = lDoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Create a new comment annotation
PdfPopupAnnotation comment = new PdfPopupAnnotation();
//Set author
comment.Author = "John";
//Set Text
comment.Text = "This is first comment";
//Set modification date
comment.ModifiedDate = DateTime.Now;
//Set subject
comment.Subject = "Annotation Comments";
//Add the comments to the annotation
loadedRectangleAnnotation.Comments.Add(comment);
//Save the document into stream
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
stream.Position = 0;
//Closes the document
lDoc.Close(true);
//Defining the Content Type for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
inputAnnotation.pdf");
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage lpage = lDoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
```

```

PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Create a new comment annotation
PdfPopupAnnotation comment = new PdfPopupAnnotation();
//Set author
comment.Author = "John";
//Set Text
comment.Text = "This is first comment";
//Set modification date
comment.ModifiedDate = DateTime.Now;
//Set subject
comment.Subject = "Annotation Comments";
//Add the comment to the annotation
loadedRectangleAnnotation.Comments.Add(comment);
//Save the document into stream
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
//Close the document
lDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Adding review status to the PDF annotation

The following code example explains how to add a review status in a newly created PDF annotation.

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create a new page
PdfPage page = document.Pages.Add();
//Create new rectangle annotation
PdfRectangleAnnotation rectangleAnnotation = new PdfRectangleAnnotation(new
RectangleF(0, 0, 100, 50), "Rectangle Annotation");
//Set author
rectangleAnnotation.Author = "Syncfusion";
rectangleAnnotation.Border.BorderWidth = 1;
rectangleAnnotation.Color = Color.Red;
rectangleAnnotation.ModifiedDate = DateTime.Now;
//Create a new review annotation
PdfPopupAnnotation review = new PdfPopupAnnotation();
//Set author
review.Author = "John";

```

```

//Set review state model
review.StateModel = PdfAnnotationStateModel.Review;
//Set review state
review.State = PdfAnnotationState.Accepted;
//Set modification date
review.ModifiedDate = DateTime.Now;
//Add the review to the annotation
rectangleAnnotation.ReviewHistory.Add(review);
//Add the annotation to the PDF page
page.Annotations.Add(rectangleAnnotation);
//Save the document to disk
document.Save("Output.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument
'Create a new page
Dim page As PdfPage = document.Pages.Add
'Create new rectangle annotation
Dim rectangleAnnotation As PdfRectangleAnnotation = New
PdfRectangleAnnotation(New RectangleF(0, 0, 100, 50), "Rectangle
Annotation")
'Set author
rectangleAnnotation.Author = "Syncfusion"
rectangleAnnotation.Border.BorderWidth = 1
rectangleAnnotation.Color = Color.Red
rectangleAnnotation.ModifiedDate = DateTime.Now
Dim review As PdfPopupAnnotation = New PdfPopupAnnotation
'Set author
review.Author = "John"
'Set review state model
review.StateModel = PdfAnnotationStateModel.Review
'Set review state
review.State = PdfAnnotationState.Accepted
'Set modification date.
review.ModifiedDate = DateTime.Now
'Add the review to the annotation.
rectangleAnnotation.ReviewHistory.Add(review)
'Add the annotation to the PDF page.
page.Annotations.Add(rectangleAnnotation)
'Save the document to disk.
document.Save("Output.pdf")
'Close the document
document.Close(true)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create a new page
PdfPage page = document.Pages.Add();
//Create new rectangle annotation

```

```

PdfRectangleAnnotation rectangleAnnotation = new PdfRectangleAnnotation(new
RectangleF(0, 0, 100, 50), "Rectangle Annotation");
//Set author
rectangleAnnotation.Author = "Syncfusion";
rectangleAnnotation.Border.BorderWidth = 1;
rectangleAnnotation.Color = Color.Red;
rectangleAnnotation.ModifiedDate = DateTime.Now;
//Create a new review annotation
PdfPopupAnnotation review = new PdfPopupAnnotation();
//Set author
review.Author = "John";
//Set review state model
review.StateModel = PdfAnnotationStateModel.Review;
//Set review state
review.State = PdfAnnotationState.Accepted;
//Set modification date
review.ModifiedDate = DateTime.Now;
//Add the review to the annotation
rectangleAnnotation.ReviewHistory.Add(review);
//Add the annotation to the PDF page
page.Annotations.Add(rectangleAnnotation);
//Save the document
MemoryStream stream = new MemoryStream();
//Save the PDF document to stream
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create a new page
PdfPage page = document.Pages.Add();
//Create new rectangle annotation
PdfRectangleAnnotation rectangleAnnotation = new PdfRectangleAnnotation(new
RectangleF(0, 0, 100, 50), "Rectangle Annotation");
//Set author
rectangleAnnotation.Author = "Syncfusion";
rectangleAnnotation.Border.BorderWidth = 1;
rectangleAnnotation.Color = Color.Red;
rectangleAnnotation.ModifiedDate = DateTime.Now;
//Create a new review annotation
PdfPopupAnnotation review = new PdfPopupAnnotation();
//Set author
review.Author = "John";
//Set review state model
review.StateModel = PdfAnnotationStateModel.Review;
//Set review state
review.State = PdfAnnotationState.Accepted;
//Set modification date
review.ModifiedDate = DateTime.Now;
//Add the review to the annotation

```

```

rectangleAnnotation.ReviewHistory.Add(review);
//Add the annotation to the PDF page
page.Annotations.Add(rectangleAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the Content Type for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "PopupAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create a new page
PdfPage page = document.Pages.Add();
//Create new rectangle annotation
PdfRectangleAnnotation rectangleAnnotation = new PdfRectangleAnnotation(new
RectangleF(0, 0, 100, 50), "Rectangle Annotation");
//Set author
rectangleAnnotation.Author = "Syncfusion";
rectangleAnnotation.Border.BorderWidth = 1;
rectangleAnnotation.Color = Color.Red;
rectangleAnnotation.ModifiedDate = DateTime.Now;
//Create a new review annotation
PdfPopupAnnotation review = new PdfPopupAnnotation();
//Set author
review.Author = "John";
//Set review state model
review.StateModel = PdfAnnotationStateModel.Review;
//Set review state
review.State = PdfAnnotationState.Accepted;
//Set modification date
review.ModifiedDate = DateTime.Now;
//Add the review to the annotation
rectangleAnnotation.ReviewHistory.Add(review);
//Add the annotation to the PDF page
page.Annotations.Add(rectangleAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)

```

```
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

The following code example explains how to add the review status to the existing PDF annotation.

C#

```
//Load the PDF document
PdfLoadedDocument ldoc = new PdfLoadedDocument("Input.pdf");
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Create a new review annotation
PdfPopupAnnotation review = new PdfPopupAnnotation();
//Set author
review.Author = "John";
//Set review state model
review.StateModel = PdfAnnotationStateModel.Review;
//Set review state
review.State = PdfAnnotationState.Accepted;
//Set modification date
review.ModifiedDate = DateTime.Now;
//Add the review to the annotation
loadedRectangleAnnotation.ReviewHistory.Add(review);
//Save the document
ldoc.Save("Output.pdf");
//Close the document
ldoc.Close(true);
```

VB.NET

```
'Load the existing PDF document
Dim ldoc As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the existing PDF page
Dim lpage As PdfLoadedPage = CType(ldoc.Pages(0), PdfLoadedPage)
'Get the existing annotations
Dim annots As PdfLoadedAnnotationCollection = lpage.Annotations
'Get the existing rectangle annotation
Dim loadedRectangleAnnotation As PdfLoadedRectangleAnnotation =
CType(annots(0), PdfLoadedRectangleAnnotation)
'Get the existing rectangle annotation
Dim review As PdfPopupAnnotation = New PdfPopupAnnotation
'Set author
review.Author = "John"
'Set review state model
```

```

review.StateModel = PdfAnnotationStateModel.Review
'Set review state
review.State = PdfAnnotationState.Accepted
'Set modification date
review.ModifiedDate = DateTime.Now
'Add the review to the annotation.
loadedRectangleAnnotation.ReviewHistory.Add(review)
'Save the document
ldoc.Save("Output.pdf")
'Close the document
ldoc.Close(true)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument lDoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await lDoc.OpenAsync(file);
//Get the existing PDF page
PdfLoadedPage lpage = lDoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Create a new review annotation
PdfPopupAnnotation review = new PdfPopupAnnotation();
//Set author
review.Author = "John";
//Set review state model
review.StateModel = PdfAnnotationStateModel.Review;
//Set review state
review.State = PdfAnnotationState.Accepted;
//Set modification date
review.ModifiedDate = DateTime.Now;
//Add the review to the annotation
loadedRectangleAnnotation.ReviewHistory.Add(review);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await lDoc.SaveAsync(stream);
//Close the document
lDoc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document

```



```

FileStream docStream = new FileStream("inputAnnotation.pdf", FileMode.Open,
    FileAccess.Read);
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage lpage = lDoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
    PdfLoadedRectangleAnnotation;
//Create a new review annotation
PdfPopupAnnotation review = new PdfPopupAnnotation();
//Set author
review.Author = "John";
//Set review state model
review.StateModel = PdfAnnotationStateModel.Review;
//Set review state
review.State = PdfAnnotationState.Accepted;
//Set modification date
review.ModifiedDate = DateTime.Now;
//Add the review to the annotation
loadedRectangleAnnotation.ReviewHistory.Add(review);
//Save the document into stream
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
stream.Position = 0;
//Closes the document
lDoc.Close(true);
//Defining the Content Type for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
    inputAnnotation.pdf");
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage lpage = lDoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
    PdfLoadedRectangleAnnotation;
//Create a new review annotation
PdfPopupAnnotation review = new PdfPopupAnnotation();
//Set author
review.Author = "John";
//Set review state model
review.StateModel = PdfAnnotationStateModel.Review;

```

```

//Set review state
review.State = PdfAnnotationState.Accepted;
//Set modification date
review.ModifiedDate = DateTime.Now;
//Add the review to the annotation
loadedRectangleAnnotation.ReviewHistory.Add(review);
//Save the document into stream
MemoryStream stream = new MemoryStream();
lDoc.Save(stream);
//Close the document
lDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Removing comments and review status from PDF annotation

The Essential PDF supports removing comments and reviewing status from the PDF annotation.

The following code example explains how to remove comments from the existing PDF annotation.

C#

```

//Load the PDF document
PdfLoadedDocument ldoc = new PdfLoadedDocument("Input.pdf");
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Get the annotation comments collection
PdfLoadedPopupAnnotationCollection commentsCollection =
loadedRectangleAnnotation.Comments;
// Remove comments by index
commentsCollection.RemoveAt(0);
//Save the document
ldoc.Save("Output.pdf");
//Close the document
ldoc.Close(true);

```

VB.NET

```
'Load the PDF document
```

```

Dim ldoc As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf") 'Load the
PDF page
'Get the existing PDF page
Dim lpage As PdfLoadedPage = CType(ldoc.Pages(0), PdfLoadedPage)
'Get the existing annotations
Dim annots As PdfLoadedAnnotationCollection = lpage.Annotations
'Get the existing rectangle annotation
Dim loadedRectangleAnnotation As PdfLoadedRectangleAnnotation
=CType(annots(0), PdfLoadedRectangleAnnotation)
'Get the annotation comments collection
Dim commentsCollection As PdfLoadedPopupAnnotationCollection =
loadedRectangleAnnotation.Comments
' Remove comments by index
commentsCollection.RemoveAt(0)
'Save the document
ldoc.Save("Output.pdf")
'Close the document
ldoc.Close(true)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument ldoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await ldoc.OpenAsync(file);
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Get the annotation comments collection
PdfLoadedPopupAnnotationCollection commentsCollection =
loadedRectangleAnnotation.Comments;
// Remove comments by index
commentsCollection.RemoveAt(0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await ldoc.SaveAsync(stream);
//Close the document
ldoc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document

```

```

FileStream docStream = new FileStream("inputAnnotation.pdf", FileMode.Open,
    FileAccess.Read);
PdfLoadedDocument ldoc = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
    PdfLoadedRectangleAnnotation;
//Get the annotation comments collection
PdfLoadedPopupAnnotationCollection commentsCollection =
    loadedRectangleAnnotation.Comments;
// Remove comments by index
commentsCollection.RemoveAt(0);
//Save the document into stream
MemoryStream stream = new MemoryStream();
ldoc.Save(stream);
stream.Position = 0;
//Closes the document
ldoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
    inputAnnotation.pdf");
PdfLoadedDocument ldoc = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
    PdfLoadedRectangleAnnotation;
//Get the annotation comments collection
PdfLoadedPopupAnnotationCollection commentsCollection =
    loadedRectangleAnnotation.Comments;
// Remove comments by index
commentsCollection.RemoveAt(0);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
{

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

The following code example explains how to remove review status to the existing PDF annotation.

C#

```
//Load the PDF document
PdfLoadedDocument ldoc = new PdfLoadedDocument("Input.pdf");
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Get the annotation review collection
PdfLoadedPopupAnnotationCollection reviewCollection =
loadedRectangleAnnotation.ReviewHistory;
// Remove review status by index
reviewCollection.RemoveAt(0);
ldoc.Save("Output.pdf");
//Close the document
ldoc.Close(true);
```

VB.NET

```
'Loaded the PDF document
Dim ldoc As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the existing PDF page
Dim lpage As PdfLoadedPage = CType(ldoc.Pages(0), PdfLoadedPage)
'Get the existing annotations
Dim annots As PdfLoadedAnnotationCollection = lpage.Annotations
'Get the existing rectangle annotation
Dim loadedRectangleAnnotation As PdfLoadedRectangleAnnotation =
CType(annots(0), PdfLoadedRectangleAnnotation)
'Get the annotation review collection
Dim reviewCollection As PdfLoadedPopupAnnotationCollection =
loadedRectangleAnnotation.ReviewHistory
' Remove review status by index
reviewCollection.RemoveAt(0)
ldoc.Save("Output.pdf")
'Close the document
ldoc.Close(true)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
```

```

picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument ldoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await ldoc.OpenAsync(file);
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
// Get the annotation review collection
PdfLoadedPopupAnnotationCollection reviewCollection =
loadedRectangleAnnotation.ReviewHistory;
//Remove review status by index
reviewCollection.RemoveAt(0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await ldoc.SaveAsync(stream);
//Close the document
ldoc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("inputAnnotation.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument ldoc = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Get the annotation reviewcollection
PdfLoadedPopupAnnotationCollection reviewCollection =
loadedRectangleAnnotation.ReviewHistory;
//Remove review status by index
reviewCollection.RemoveAt(0);
//Save the document into stream
MemoryStream stream = new MemoryStream();
ldoc.Save(stream);
stream.Position = 0;
//Closes the document
ldoc.Close(true);
//Defining the Content Type for pdf file
string contentType = "application/pdf";
//Define the file name

```

```
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
inputAnnotation.pdf");
PdfLoadedDocument ldoc = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Get the annotation review collection
PdfLoadedPopupAnnotationCollection reviewCollection =
loadedRectangleAnnotation.ReviewHistory;
//Remove review status by index
reviewCollection.RemoveAt(0);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Modifying comments and review status

The Essential PDF supports modifying comments and reviewing status in the PDF annotation.

The following code example explains how to modify comments in the existing PDF annotation.

C#

```
//Load the PDF document
PdfLoadedDocument ldoc = new PdfLoadedDocument("Input.pdf");
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
```

```

//Get the annotation comments collection
PdfLoadedPopupAnnotationCollection commentsCollection =
loadedRectangleAnnotation.Comments;
//Get the modified comments
PdfLoadedPopupAnnotation loadedComments = commentsCollection[0];
//Modify the comments Text
loadedComments.Text = "This is the modified comment";
//Save the document
ldoc.Save("Output.pdf");
//Close the document
ldoc.Close(true);

```

VB.NET

```

'Load the PDF document
Dim ldoc As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the existing PDF page
Dim lpage As PdfLoadedPage = CType(ldoc.Pages(0), PdfLoadedPage)
'Get the existing annotations
Dim annots As PdfLoadedAnnotationCollection = lpage.Annotations
'Get the existing rectangle annotation
Dim loadedRectangleAnnotation As PdfLoadedRectangleAnnotation =
CType(annots(0), PdfLoadedRectangleAnnotation)
'Get the annotation comments collection
Dim commentsCollection As PdfLoadedPopupAnnotationCollection =
loadedRectangleAnnotation.Comments
'Get the modified comment
Dim loadedComments As PdfLoadedPopupAnnotation = commentsCollection(0)
' Modify the comment Text
loadedComments.Text = "This is the modified comment"
'Save the document
ldoc.Save("Output.pdf")
'Close the document
ldoc.Close(true)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument ldoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await ldoc.OpenAsync(file);
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Load the annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as PdfLoadedRectangleAnnotation;
//Get the existing rectangle annotation

```



```

PdfLoadedPopupAnnotationCollection commentsCollection =
loadedRectangleAnnotation.Comments;
// Get the modified comment
PdfLoadedPopupAnnotation loadedComments = commentsCollection[0];
// Modify the comment Text
loadedComments.Text = "This is the modified comment";
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await ldoc.SaveAsync(stream);
//Close the document
ldoc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("inputAnnotation.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument ldoc = new PdfLoadedDocument(docStream);
//Load the PDF document page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Load the annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Get the annotation comments collection
PdfLoadedPopupAnnotationCollection commentsCollection =
loadedRectangleAnnotation.Comments;
//Get the modified comment
PdfLoadedPopupAnnotation loadedComments = commentsCollection[0];
//Modify the comment Text
loadedComments.Text = "This is the modified comments";
//Save the document into stream
MemoryStream stream = new MemoryStream();
ldoc.Save(stream);
stream.Position = 0;
//Closes the document
ldoc.Close(true);
//Defining the Content Type for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
inputAnnotation.pdf");
PdfLoadedDocument ldoc = new PdfLoadedDocument(docStream);
//Load the PDF document page

```

```

PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Load the annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Get the annotation comments collection
PdfLoadedPopupAnnotationCollection commentsCollection =
loadedRectangleAnnotation.Comments;
//Get the modified comment
PdfLoadedPopupAnnotation loadedComments = commentsCollection[0];
//Modify the comments Text
loadedComments.Text = "This is Modify comments";
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following code example explains how to modify review status to the existing PDF annotation.

C#

```

//Load the PDF document
PdfLoadedDocument ldoc = new PdfLoadedDocument("Input.pdf");
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Get the annotation review collection
PdfLoadedPopupAnnotationCollection reviewCollection =
loadedRectangleAnnotation.ReviewHistory;
// Get the modified review state
PdfLoadedPopupAnnotation loadedReview = reviewCollection[0];
// Modify the review State
loadedReview.State = PdfAnnotationState.Rejected;
//Save the document
ldoc.Save("Output.pdf");
//Close the document
ldoc.Close(true);

```

VB.NET

```
'Loaded the PDF document
```

```

Dim ldoc As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the existing PDF page
Dim lpage As PdfLoadedPage = CType(ldoc.Pages(0), PdfLoadedPage)
'Get the existing annotations
Dim annots As PdfLoadedAnnotationCollection = lpage.Annotations
'Get the existing rectangle annotation
Dim loadedRectangleAnnotation As PdfLoadedRectangleAnnotation =
CType(annots(0), PdfLoadedRectangleAnnotation)
'Get annotation review collection
Dim reviewCollection As PdfLoadedPopupAnnotationCollection =
loadedRectangleAnnotation.ReviewHistory
Dim loadedReview As PdfLoadedPopupAnnotation = reviewCollection(0)
' Modify the review State
loadedReview.State = PdfAnnotationState.Rejected
'Save the document
ldoc.Save("Output.pdf")
'Close the document
ldoc.Close(true)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument ldoc = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await ldoc.OpenAsync(file);
//Get the existing annotations
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing PDF page
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Get the annotation review collection
PdfLoadedPopupAnnotationCollection reviewCollection =
loadedRectangleAnnotation.ReviewHistory;
// Get the modified review state
PdfLoadedPopupAnnotation loadedReview = reviewCollection[0];
// Modify the review State
loadedReview.State = PdfAnnotationState.Rejected;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await ldoc.SaveAsync(stream);
//Close the document
ldoc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("inputAnnotation.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument ldoc = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Get the annotation review collection
PdfLoadedPopupAnnotationCollection reviewCollection =
loadedRectangleAnnotation.ReviewHistory;
// Get the modified review state
PdfLoadedPopupAnnotation loadedReview = reviewCollection[0];
// Modify the review State
loadedReview.State = PdfAnnotationState.Rejected;
//Save the document into stream
MemoryStream stream = new MemoryStream();
ldoc.Save(stream);
stream.Position = 0;
//Closes the document
ldoc.Close(true);
//Defining the Content Type for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
inputAnnotation.pdf");
PdfLoadedDocument ldoc = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage lpage = ldoc.Pages[0] as PdfLoadedPage;
//Get the existing annotations
PdfLoadedAnnotationCollection annots = lpage.Annotations;
//Get the existing rectangle annotation
PdfLoadedRectangleAnnotation loadedRectangleAnnotation = annots[0] as
PdfLoadedRectangleAnnotation;
//Get the annotation review collection
PdfLoadedPopupAnnotationCollection reviewCollection =
loadedRectangleAnnotation.ReviewHistory;
// Get the modified review state
PdfLoadedPopupAnnotation loadedReview = reviewCollection[0];
// Modify the review State
loadedReview.State = PdfAnnotationState.Rejected;
//Save the stream into pdf file
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Retrieve review status and comments from PDF annotation

The PDF annotations may have an author-specific state associated with them. The state is not specified in the annotation itself, but it represents a separate text annotation ([Popup Annotation](#)).

The Essential PDF supports retrieving the annotation comments and review history from the existing PDF document annotations.

Retrieve review status from PDF annotation

You can retrieve the annotation review history from the existing PDF document annotations through the [ReviewHistory](#) property.

C#

```
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("input.pdf");
//Get the existing PDF page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the annotation
PdfLoadedTextMarkupAnnotation loadedMarkup = loadedPage.Annotations[0] as
PdfLoadedTextMarkupAnnotation;
//Get the review history collection for the annotation
PdfLoadedPopupAnnotationCollection reviewCollection =
loadedMarkup.ReviewHistory;
//Get annotation state
PdfAnnotationState state = reviewCollection[0].State;
//Get annotation state model
PdfAnnotationStateModel model = reviewCollection[0].StateModel;
//Get the comments of the annotation
PdfLoadedPopupAnnotationCollection commentsCollection =
loadedMarkup.Comments;
//Get the review history of the comment
PdfLoadedPopupAnnotationCollection reviewCollection1 =
commentsCollection[0].ReviewHistory;
//Close the PDF document
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("input.pdf")
```

```

'Get the existing PDF page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Get the annotation
Dim loadedMarkup As PdfLoadedTextMarkupAnnotation =
TryCast(loadedPage.Annotations(0), PdfLoadedTextMarkupAnnotation)
'Get the review history collection for the annotation
Dim reviewCollection As PdfLoadedPopupAnnotationCollection =
loadedMarkup.ReviewHistory
'Get annotation state
Dim state As PdfAnnotationState = reviewCollection(0).State
'Get annotation state model
Dim model As PdfAnnotationStateModel = reviewCollection(0).StateModel
'Get the comments of the annotation
Dim commentsCollection As PdfLoadedPopupAnnotationCollection =
loadedMarkup.Comments
'Get the review history of the comment
Dim reviewCollection1 As PdfLoadedPopupAnnotationCollection =
commentsCollection(0).ReviewHistory
'Close the PDF document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the existing PDF page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the annotation
PdfLoadedTextMarkupAnnotation loadedMarkup = loadedPage.Annotations[0] as
PdfLoadedTextMarkupAnnotation;
//Get the review history collection for the annotation
PdfLoadedPopupAnnotationCollection reviewCollection =
loadedMarkup.ReviewHistory;
//Get annotation state
PdfAnnotationState state = reviewCollection[0].State;
//Get annotation state model
PdfAnnotationStateModel model = reviewCollection[0].StateModel;
//Get the comments of the annotation
PdfLoadedPopupAnnotationCollection commentsCollection =
loadedMarkup.Comments;
//Get the review history of the comment
PdfLoadedPopupAnnotationCollection reviewCollection1 =
commentsCollection[0].ReviewHistory;
//Close the document
loadedDocument.Close(true);

```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the annotation
PdfLoadedTextMarkupAnnotation loadedMarkup = loadedPage.Annotations[0] as
PdfLoadedTextMarkupAnnotation;
//Get the review history collection for the annotation
PdfLoadedPopupAnnotationCollection reviewCollection =
loadedMarkup.ReviewHistory;
//Get annotation state
PdfAnnotationState state = reviewCollection[0].State;
//Get annotation state model
PdfAnnotationStateModel model = reviewCollection[0].StateModel;
//Get the comments of the annotation
PdfLoadedPopupAnnotationCollection commentsCollection =
loadedMarkup.Comments;
//Get the review history of the comment
PdfLoadedPopupAnnotationCollection reviewCollection1 =
commentsCollection[0].ReviewHistory;
//Closes the document
loadedDocument.Close(true);
```

XAMARIN

```
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the annotation
PdfLoadedTextMarkupAnnotation loadedMarkup = loadedPage.Annotations[0] as
PdfLoadedTextMarkupAnnotation;
//Get the review history collection for the annotation
PdfLoadedPopupAnnotationCollection reviewCollection =
loadedMarkup.ReviewHistory;
//Get annotation state
PdfAnnotationState state = reviewCollection[0].State;
//Get annotation state model
PdfAnnotationStateModel model = reviewCollection[0].StateModel;
//Get the comments of the annotation
PdfLoadedPopupAnnotationCollection commentsCollection =
loadedMarkup.Comments;
//Get the review history of the comment
PdfLoadedPopupAnnotationCollection reviewCollection1 =
commentsCollection[0].ReviewHistory;
//Closes the document
loadedDocument.Close(true);
```

Retrieve comments from PDF annotation

The following code example explains how to retrieve the annotation comments from the existing PDF document annotations.

C#

```
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("input.pdf");
//Get the existing PDF page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the annotation
PdfLoadedTextMarkupAnnotation loadedMarkup = loadedPage.Annotations[0] as
PdfLoadedTextMarkupAnnotation;
//Get the comments of the annotation
PdfLoadedPopupAnnotationCollection commentsCollection =
loadedMarkup.Comments;
//Close the PDF document
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("input.pdf")
'Get the existing PDF page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Get the annotation
Dim loadedMarkup As PdfLoadedTextMarkupAnnotation =
TryCast(loadedPage.Annotations(0), PdfLoadedTextMarkupAnnotation)
'Get the comments of the annotation
Dim commentsCollection As PdfLoadedPopupAnnotationCollection =
loadedMarkup.Comments
'Close the PDF document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and choose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Get the existing PDF page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the annotation
PdfLoadedTextMarkupAnnotation loadedMarkup = loadedPage.Annotations[0] as
PdfLoadedTextMarkupAnnotation;
//Get the comments of the annotation
PdfLoadedPopupAnnotationCollection commentsCollection =
loadedMarkup.Comments;
//Close the document
```



```
loadedDocument.Close(true);
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument lDoc = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the annotation
PdfLoadedTextMarkupAnnotation loadedMarkup = loadedPage.Annotations[0] as
PdfLoadedTextMarkupAnnotation;
//Get the comments of the annotation
PdfLoadedPopupAnnotationCollection commentsCollection =
loadedMarkup.Comments;
//Closes the document
loadedDocument.Close(true);
```

XAMARIN

```
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the existing PDF page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Get the annotation
PdfLoadedTextMarkupAnnotation loadedMarkup = loadedPage.Annotations[0] as
PdfLoadedTextMarkupAnnotation;
//Get the comments of the annotation
PdfLoadedPopupAnnotationCollection commentsCollection =
loadedMarkup.Comments;
//Closes the document
loadedDocument.Close(true);
```

Printing Annotations

The Essential PDF supports printing the annotation in a PDF document by setting the annotation flag to **Print** using the [AnnotationFlags](#) property.

The following code example illustrates how to print annotation in the PDF document.

C#

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new PDF rubber stamp annotation
RectangleF rectangle = new RectangleF(40, 60, 80, 20);
PdfRubberStampAnnotation rubberStampAnnotation = new
PdfRubberStampAnnotation(rectangle, "Text Rubber Stamp Annotation");
rubberStampAnnotation.Icon = PdfRubberStampAnnotationIcon.Draft;
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation";
//Set the AnnotationFlags to print
```

```

rubberStampAnnotation.AnnotationFlags = PdfAnnotationFlags.Print;
//Adds annotation to the page
page.Annotations.Add(rubberStampAnnotation);
//Saves the document
document.Save("RubberStamp.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Creates a new PDF document
Dim document As PdfDocument = New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Creates a new PDF rubber stamp annotation
Dim rectangle As RectangleF = New RectangleF(40, 60, 80, 20)
Dim rubberStampAnnotation As PdfRubberStampAnnotation = New
PdfRubberStampAnnotation(rectangle, " Text Rubber Stamp Annotation")
rubberStampAnnotation.Icon = PdfRubberStampAnnotationIcon.Draft
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation"
'Set the AnnotationFlags to print
rubberStampAnnotation.AnnotationFlags = PdfAnnotationFlags.Print
'Adds annotation to the page
page.Annotations.Add(rubberStampAnnotation)
'Saves the document
document.Save("RubberStamp.pdf")
'Close the document
document.Close(True)

```

UWP

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new PDF rubber stamp annotation
RectangleF rectangle = new RectangleF(40, 60, 80, 20);
PdfRubberStampAnnotation rubberStampAnnotation = new
PdfRubberStampAnnotation(rectangle, " Text Rubber Stamp Annotation");
rubberStampAnnotation.Icon = PdfRubberStampAnnotationIcon.Draft;
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation";
//Set the AnnotationFlags to print
rubberStampAnnotation.AnnotationFlags = PdfAnnotationFlags.Print;
//Adds annotation to the page
page.Annotations.Add(rubberStampAnnotation);
//Saves the document
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new PDF rubber stamp annotation
RectangleF rectangle = new RectangleF(40, 60, 80, 20);
PdfRubberStampAnnotation rubberStampAnnotation = new
PdfRubberStampAnnotation(rectangle, "Text Rubber Stamp Annotation");
rubberStampAnnotation.Icon = PdfRubberStampAnnotationIcon.Draft;
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation";
//Set the AnnotationFlags to print
rubberStampAnnotation.AnnotationFlags = PdfAnnotationFlags.Print;
//Adds annotation to the page
page.Annotations.Add(rubberStampAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "RubberStamp.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Creates a new PDF rubber stamp annotation
RectangleF rectangle = new RectangleF(40, 60, 80, 20);
PdfRubberStampAnnotation rubberStampAnnotation = new
PdfRubberStampAnnotation(rectangle, "Text Rubber Stamp Annotation");
rubberStampAnnotation.Icon = PdfRubberStampAnnotationIcon.Draft;
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation";
//Set the AnnotationFlags to print
rubberStampAnnotation.AnnotationFlags = PdfAnnotationFlags.Print;
//Adds annotation to the page
page.Annotations.Add(rubberStampAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
```

```

Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("RubberStamp.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("RubberStamp.pdf", "application/pdf", stream);
}

```

The following table explains annotation flags.

Member	Meaning
Invisible	If set, do not display the annotation if it does not belong to one of the standard annotation types and no annotation handler is available.
Hidden	If set, do not display or print the annotation, or allow user interact with annotation, regardless of annotation type or annotation handler.
Print	If set, prints the annotation when the page is printed.
NoZoom	If set, do not scale the annotation's appearance to match the magnification of the page.
NoRotate	If set, do not rotate the annotation's appearance to match the rotation of the page.
NoView	If set, do not display the annotation on the screens or allow user interact with annotation.
ReadOnly	If set, do not allow user interact with annotation.
Locked	If set, do not allow the annotation to be deleted or its properties to be modified by the user.
ToggleNoView	If set, inverts the interpretation of the NoView flag for certain events.

Add Custom Stamp using Rubber Stamp Annotation

Essential PDF supports adding custom stamp in an existing PDF document by using the [PdfRubberStampAnnotation](#) class along with different appearance settings through [PdfAppearance](#). This custom stamp is movable and resizable.

Rubber stamp annotation displays text or graphics intended to look like it is stamped on the page with a rubber stamp. When opened, it displays a pop-up window containing the text of the associated note.

The following code snippet explains how to add custom stamp in an existing PDF document using rubber stamp annotation.

C#

```

//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get the page from loaded PDF document
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a new pdf rubber stamp annotation
RectangleF rectangleF = new RectangleF(350, 20, 200, 80);
PdfRubberStampAnnotation rubberStampAnnotation = new
PdfRubberStampAnnotation(rectangleF);
//Custom stamp the rubber stamp annotation
PdfSolidBrush brush = new PdfSolidBrush(new PdfColor(Color.LightBlue));
PdfPath path = RoundedRect(new RectangleF(0, 0, 200, 80), 20);
rubberStampAnnotation.Appearance.Normal.Graphics.DrawPath(brush, path);
//Add text in rubber stamp annotation
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Bold);
rubberStampAnnotation.Appearance.Normal.Graphics.DrawString("DD/2018/1234567
890", font, PdfBrushes.Black, new PointF(10, 20));
rubberStampAnnotation.Appearance.Normal.Graphics.DrawString(DateTime.Now.ToS
tring("dd-MM-yyyy HH:mm:ss"), font, PdfBrushes.Black, new PointF(10, 40));
//Set the content of annotation
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation";
//Add annotation to the page
loadedPage.Annotations.Add(rubberStampAnnotation);
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
public static PdfPath RoundedRect(RectangleF bounds, int radius)
{
    int diameter = radius * 2;
    SizeF size = new SizeF(diameter, diameter);
    RectangleF arc = new RectangleF(bounds.Location, size);
    PdfPath path = new PdfPath();
    if (radius == 0)
    {
        path.AddRectangle(bounds);
        return path;
    }
    //Draw the top left arc
    path.AddArc(arc, 180, 90);
    //Draw the top right arc
    arc.X = bounds.Right - diameter;
    path.AddArc(arc, 270, 90);
    //Draw the bottom right arc
    arc.Y = bounds.Bottom - diameter;
    path.AddArc(arc, 0, 90);
    //Draw the bottom left arc
    arc.X = bounds.Left;
    path.AddArc(arc, 90, 90);
    //Close the figure
    path.CloseFigure();
    //Return the path
    return path;
}

```

VB.NET

```

'Load an existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the page from loaded PDF document
Dim loadedPage As PdfLoadedPage = CType(loadedDocument.Pages(0), PdfLoadedPage)
'Create a new pdf rubber stamp annotation
Dim rectangleF As RectangleF = New RectangleF(350, 20, 200, 80)
Dim rubberStampAnnotation As PdfRubberStampAnnotation = New PdfRubberStampAnnotation(rectangleF)
'Custom stamp the rubber stamp annotation
Dim brush As PdfSolidBrush = New PdfSolidBrush(New PdfColor(Color.LightBlue))
Dim path As PdfPath = RoundedRect(New RectangleF(0, 0, 200, 80), 20)
rubberStampAnnotation.Appearance.Normal.Graphics.DrawPath(brush, path)
'Add text in rubber stamp annotation
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold)
rubberStampAnnotation.Appearance.Normal.Graphics.DrawString("DD/2018/1234567890", font, PdfBrushes.Black, New PointF(10, 20))
rubberStampAnnotation.Appearance.Normal.Graphics.DrawString(DateTime.Now.ToString("dd-MM-yyyy HH:mm:ss"), font, PdfBrushes.Black, New PointF(10, 40))
'Set the content of annotation
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation"
'Adds annotation to the page
loadedPage.Annotations.Add(rubberStampAnnotation)
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the instance of PdfLoadedDocument
loadedDocument.Close(True)
Private Function RoundedRect(bounds As RectangleF, radius As Integer) As PdfPath
Dim diameter As Integer = (radius * 2)
Dim size As.SizeF = New.SizeF(diameter, diameter)
Dim arc As RectangleF = New.RectangleF(bounds.Location, size)
Dim path As PdfPath = New PdfPath
If (radius = 0) Then
path.AddRectangle(bounds)
Return path
End If
'Draw the top left arc
path.AddArc(arc, 180, 90)
'Draw the top right arc
arc.X = (bounds.Right - diameter)
path.AddArc(arc, 270, 90)
'Draw the bottom right arc
arc.Y = (bounds.Bottom - diameter)
path.AddArc(arc, 0, 90)
'Draw the bottom left arc
arc.X = bounds.Left
path.AddArc(arc, 90, 90)
'Close the figure
path.CloseFigure()
'Return the path
Return path
End Function

```

UWP

```

//Load an existing PDF document
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page from loaded PDF document
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a new pdf rubber stamp annotation
RectangleF rectangleF = new RectangleF(350, 20, 200, 80);
PdfRubberStampAnnotation rubberStampAnnotation = new
PdfRubberStampAnnotation(rectangleF);
//Custom stamp the rubber stamp annotation
PdfSolidBrush brush = new PdfSolidBrush(new PdfColor(Color.FromArgb(255,
173, 216, 230)));
PdfPath path = RoundedRect(new RectangleF(0, 0, 200, 80), 20);
rubberStampAnnotation.Appearance.Normal.Graphics.DrawPath(brush, path);
//Add text in rubber stamp annotation
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Bold);
rubberStampAnnotation.Appearance.Normal.Graphics.DrawString("DD/2018/1234567
890", font, PdfBrushes.Black, new PointF(10, 20));
rubberStampAnnotation.Appearance.Normal.Graphics.DrawString(DateTime.Now.ToS
tring("dd-MM-yyyy HH:mm:ss"), font, PdfBrushes.Black, new PointF(10, 40));
//Set the content of annotation
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation";
//Add annotation to the page
loadedPage.Annotations.Add(rubberStampAnnotation);
//Create memory stream
MemoryStream ms = new MemoryStream();
//Open the document in browser after saving it
loadedDocument.Save(ms);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code sample
Save(ms, "Output.pdf");
private PdfPath RoundedRect(RectangleF bounds, int radius)
{
    int diameter = radius * 2;
   .SizeF size = new SizeF(diameter, diameter);
    RectangleF arc = new RectangleF(bounds.Location, size);
    PdfPath path = new PdfPath();
    if (radius == 0)
    {
        path.AddRectangle(bounds);
        return path;
    }
    //Draw the top left arc
    path.AddArc(arc, 180, 90);
    //Draw the top right arc
    arc.X = bounds.Right - diameter;
    path.AddArc(arc, 270, 90);
    //Draw the bottom right arc
    arc.Y = bounds.Bottom - diameter;
    path.AddArc(arc, 0, 90);
}

```

```
//Draw the bottom left arc
arc.X = bounds.Left;
path.AddArc(arc, 90, 90);
//Close the figure
path.CloseFigure();
//Return the path
return path;
}
```

ASP.NET CORE

```
//Load an existing PDF document
FileStream inputStream = new FileStream("Input.pdf", FileMode.Open);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page from loaded PDF document
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a new pdf rubber stamp annotation
RectangleF rectangleF = new RectangleF(350, 20, 200, 80);
PdfRubberStampAnnotation rubberStampAnnotation = new
PdfRubberStampAnnotation(rectangleF);
//Custom stamp the rubber stamp annotation
PdfSolidBrush brush = new PdfSolidBrush(new PdfColor(Color.FromArgb(255,
173, 216, 230)));
PdfPath path = RoundedRect(new RectangleF(0, 0, 200, 80), 20);
rubberStampAnnotation.Appearance.Normal.Graphics.DrawPath(brush, path);
//Add text in rubber stamp annotation
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Bold);
rubberStampAnnotation.Appearance.Normal.Graphics.DrawString("DD/2018/1234567
890", font, PdfBrushes.Black, new PointF(10, 20));
rubberStampAnnotation.Appearance.Normal.Graphics.DrawString(DateTime.Now.ToS
tring("dd-MM-yyyy HH:mm:ss"), font, PdfBrushes.Black, new PointF(10, 40));
//Set the content of annotation
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation";
//Add annotation to the page
loadedPage.Annotations.Add(rubberStampAnnotation);
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
private PdfPath RoundedRect(RectangleF bounds, int radius)
{
int diameter = radius * 2;
SizeF size = new SizeF(diameter, diameter);
RectangleF arc = new RectangleF(bounds.Location, size);
PdfPath path = new PdfPath();
if (radius == 0)
{
path.AddRectangle(bounds);
return path;
}
```



```

}
//Draw the top left arc
path.AddArc(arc, 180, 90);
//Draw the top right arc
arc.X = bounds.Right - diameter;
path.AddArc(arc, 270, 90);
//Draw the bottom right arc
arc.Y = bounds.Bottom - diameter;
path.AddArc(arc, 0, 90);
//Draw the bottom left arc
arc.X = bounds.Left;
path.AddArc(arc, 90, 90);
//Close the figure
path.CloseFigure();
//Return the path
return path;
}

```

XAMARIN

```

//Load an existing PDF document
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the page from loaded PDF document
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a new pdf rubber stamp annotation
RectangleF rectangleF = new RectangleF(350, 20, 200, 80);
PdfRubberStampAnnotation rubberStampAnnotation = new
PdfRubberStampAnnotation(rectangleF);
//Custom stamp the rubber stamp annotation
PdfSolidBrush brush = new PdfSolidBrush(new
PdfColor(Syncfusion.Drawing.Color.FromArgb(255, 173, 216, 230)));
PdfPath path = RoundedRect(new RectangleF(0, 0, 200, 80), 20);
rubberStampAnnotation.Appearance.Normal.Graphics.DrawPath(brush, path);
//Add text in rubber stamp annotation
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Bold);
rubberStampAnnotation.Appearance.Normal.Graphics.DrawString("DD/2018/1234567
890", font, PdfBrushes.Black, new PointF(10, 20));
rubberStampAnnotation.Appearance.Normal.Graphics.DrawString(DateTime.Now.ToS
tring("dd-MM-yyyy HH:mm:ss"), font, PdfBrushes.Black, new PointF(10, 40));
//Set the content of annotation
rubberStampAnnotation.Text = "Text Properties Rubber Stamp Annotation";
//Add annotation to the page
loadedPage.Annotations.Add(rubberStampAnnotation);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
private PdfPath RoundedRect(RectangleF bounds, int radius)
{
int diameter = radius * 2;
SizeF size = new SizeF(diameter, diameter);
RectangleF arc = new RectangleF(bounds.Location, size);
PdfPath path = new PdfPath();
if (radius == 0)
{
path.AddRectangle(bounds);
return path;
}
//Draw the top left arc
path.AddArc(arc, 180, 90);
//Draw the top right arc
arc.X = bounds.Right - diameter;
path.AddArc(arc, 270, 90);
//Draw the bottom right arc
arc.Y = bounds.Bottom - diameter;
path.AddArc(arc, 0, 90);
//Draw the bottom left arc
arc.X = bounds.Left;
path.AddArc(arc, 90, 90);
//Close the figure
path.CloseFigure();
//Return the path
return path;
}

```

Working with Attachments

Essential PDF provides support for file attachments in PDF documents.

Attachments can contain any kind of file with detailed information.

Adding attachment to a PDF document

You can add a text file attachment to a PDF document using [PdfAttachment](#) class. The following code example illustrates this.

C#

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates an attachment
PdfAttachment attachment = new PdfAttachment("Input.txt");
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";

```

```
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Saves and closes the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Creates an attachment
Dim attachment As New PdfAttachment("Input.txt")
attachment.ModificationDate = DateTime.Now
attachment.Description = "Input.txt"
attachment.MimeType = "application/txt"
'Adds the attachment to the document
document.Attachments.Add(attachment)
'Saves and closes the PDF document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Load the file as stream
Stream fileStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.txt");
//Creates an attachment
PdfAttachment attachment = new PdfAttachment(@"Input.txt", fileStream);
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to pdf/uwp
section for respected code samples.
Save(memoryStream, "Output.pdf");
```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
Stream fileStream = new FileStream("Input.txt", FileMode.Open,
FileAccess.Read);
//Creates an attachment
PdfAttachment attachment = new PdfAttachment("Input.txt", fileStream);
attachment.ModificationDate = DateTime.Now;
```

```

attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates an attachment
Stream fileStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.txt");
PdfAttachment attachment = new PdfAttachment("Input.txt", fileStream);
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream into pdf file
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer pdf/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}

```

Essential PDF also provides support for adding the attachments to existing PDF documents. The following code example illustrates the same.

C#

```
//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Create an attachment
PdfAttachment attachment = new PdfAttachment("Input.txt");
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
if (loadedDocument.Attachments == null)
loadedDocument.CreateAttachment();
//Add the attachment to the document
loadedDocument.Attachments.Add(attachment);
//Saves and closes the PDF document
loadedDocument.Save("Output.pdf");
loadedDocument.Close(true);
```

VB.NET

```
'Load an existing PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Create an attachment
Dim attachment As New PdfAttachment("Input.txt")
attachment.ModificationDate = DateTime.Now
attachment.Description = "Input.txt"
attachment.MimeType = "application/txt"
If loadedDocument.Attachments Is Nothing Then
loadedDocument.CreateAttachment()
End If
'Add the attachment to the document
loadedDocument.Attachments.Add(attachment)
'Saves and closes the PDF document
loadedDocument.Save("Output.pdf")
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Load the file as stream
Stream fileStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.txt");
//Creates an attachment
PdfAttachment attachment = new PdfAttachment("Input.txt", fileStream);
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
```

```

if (loadedDocument.Attachments == null)
loadedDocument.CreateAttachment();
//Add the attachment to the document
loadedDocument.Attachments.Add(attachment);
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await loadedDocument.SaveAsync(memoryStream);
//Close the documents.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to pdf/uwp
section for respected code samples.
Save(memoryStream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
Stream fileStream = new FileStream("Input.txt", FileMode.Open,
FileAccess.Read);
//Creates an attachment
PdfAttachment attachment = new PdfAttachment("Input.txt", fileStream);
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
if (loadedDocument.Attachments == null)
loadedDocument.CreateAttachment();
//Add the attachment to the document
loadedDocument.Attachments.Add(attachment);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the documents.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Creates an attachment
Stream fileStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.txt");
PdfAttachment attachment = new PdfAttachment("Input.txt", fileStream);

```

```

attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
if (loadedDocument.Attachments == null)
loadedDocument.CreateAttachment();
//Add the attachment to the document
loadedDocument.Attachments.Add(attachment);
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
loadedDocument.Save(memoryStream);
//Close the documents.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer pdf/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}

```

Removing attachment from an existing PDF

Essential PDF allows you to remove the attachments from the existing document by using [Remove](#) method, as shown in the following code example.

C#

```

//Loads the PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//Removes an attachment
PdfAttachment attachment = document.Attachments[0];
document.Attachments.Remove(attachment);
document.Attachments.RemoveAt(1);
//Saves and closes the document
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Loads the PDF document
Dim document As New PdfLoadedDocument("Input.pdf")
'Removes an attachments
Dim attachment As PdfAttachment = document.Attachments(0)
document.Attachments.Remove(attachment)
document.Attachments.RemoveAt(1)
'Saves and closes the document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await document.OpenAsync(file);
//Removes an attachment
PdfAttachment attachment = document.Attachments[0];
document.Attachments.Remove(attachment);
document.Attachments.RemoveAt(1);
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to pdf/uwp section for respected code samples.
Save(memoryStream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open, FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Removes an attachment
PdfAttachment attachment = document.Attachments[0];
//document.Attachments.Remove(attachment);
document.Attachments.RemoveAt(0);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream

```



```

Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Removes an attachment
PdfAttachment attachment = document.Attachments[0];
document.Attachments.Remove(attachment);
//document.Attachments.RemoveAt(1);
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream into pdf file
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer pdf/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}

```

Extracting and saving an attachment to the disk.

Essential PDF provides support for extracting the attachments and saving them to the disk. The following code example explains how to extract and save an attachment.

C#

```

//Loads the PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Sample.pdf");
//Iterates the attachments
foreach (PdfAttachment attachment in document.Attachments)
{
//Extracts the attachment and saves it to the disk
FileStream s = new FileStream(attachment.FileName, FileMode.Create);
s.Write(attachment.Data, 0, attachment.Data.Length);
s.Dispose();
}
//Saves and closes the document
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Loads the PDF document
Dim document As New PdfLoadedDocument("Sample.pdf")
'Iterates the attachments
For Each attachment As PdfAttachment In document.Attachments

```

```
'Extracts the attachment and saves it to the disk
Dim s As New FileStream(attachment.FileName, FileMode.Create)
s.Write(attachment.Data, 0, attachment.Data.Length)
s.Dispose()
Next
'Saves and closes the document
document.Save("Output.pdf")
document.Close(True)
```

UWP

//PDF supports extracting and saving an attachment to the disk only in Windows Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Core platforms.

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Output.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Iterates the attachments
foreach (PdfAttachment attachment in document.Attachments)
{
//Extracts the attachment and saves it to the disk
FileStream s = new FileStream(attachment.FileName, FileMode.Create);
s.Write(attachment.Data, 0, attachment.Data.Length);
s.Dispose();
}
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

//PDF supports extracting and saving an attachment to the disk only in Windows Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Core platforms.

Working with Security

Essential PDF allows you to protect the PDF document using encryption and set permission to the PDF document operations like printing, editing, copy content etc. using user password and owner password. Two types of encryption algorithms are available

1. Rivest Cipher 4 (RC4)

2. Advanced Encryption Standard (AES)

Working with RC4 Encryption

You can encrypt PDF document using RC4 algorithm with 40bit or 128bit key size. The following code snippet illustrates how to encrypt the PDF document with the [UserPassword](#).

User password: Prevents people from opening or viewing a PDF document. Once the User Password is set, to open the PDF document, Adobe Acrobat/Reader will prompt a user to enter this password. If it is not correct, the document will not open. By setting a PDF User password, you can secure the PDF document.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key128Bit;
security.Algorithm = PdfEncryptionAlgorithm.RC4;
security.UserPassword = "password";
graphics.DrawString("Encrypted with RC4 128bit", font, brush, new PointF(0,
40));
//Save and close the document.
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
Dim font As New PdfStandardFont(PdfFontFamily.TimesRoman, 20.0F,
PdfFontStyle.Bold)
Dim brush As PdfBrush = PdfBrushes.Black
'Document security.
Dim security As PdfSecurity = document.Security
'Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key128Bit
security.Algorithm = PdfEncryptionAlgorithm.RC4
security.UserPassword = "password"
graphics.DrawString("Encrypted with RC4 128bit", font, brush, New PointF(0,
40))
'Save and close the document.
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key128Bit;
security.Algorithm = PdfEncryptionAlgorithm.RC4;
security.UserPassword = "password";
graphics.DrawString("Encrypted with RC4 128bit", font, brush, new PointF(0,
40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key128Bit;
security.Algorithm = PdfEncryptionAlgorithm.RC4;
security.UserPassword = "password";
graphics.DrawString("Encrypted with RC4 128bit", font, brush, new PointF(0,
40));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key128Bit;
security.Algorithm = PdfEncryptionAlgorithm.RC4;
security.UserPassword = "password";
graphics.DrawString("Encrypted with RC4 128bit", font, brush, new PointF(0,
40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

Note: While using both user and owner passwords, please specify different user and owner password while encrypting the PDF document for better security.

You can protect the PDF document from printing, editing, copying with the [OwnerPassword](#) by using the following code snippet.

Owner password: Sets PDF document restrictions, which can include printing, content copying, editing, page extracting, commenting, and more. Once the owner password is set, Acrobat will require this password to make any changes to the PDF document. It further secures the PDF document to set a PDF Owner Password.

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;

```

```

PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm.
security.KeySize = PdfEncryptionKeySize.Key128Bit;
security.Algorithm = PdfEncryptionAlgorithm.RC4;
security.OwnerPassword = "syncfusion";
//It allows printing and accessibility copy content
security.Permissions = PdfPermissionsFlags.Print |
PdfPermissionsFlags.AccessibilityCopyContent;
security.UserPassword = "password";
graphics.DrawString("This document is protected with owner password", font,
brush, new PointF(0, 40));
//Save and close the document.
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
Dim font As New PdfStandardFont(PdfFontFamily.TimesRoman, 20.0F,
PdfFontStyle.Bold)
Dim brush As PdfBrush = PdfBrushes.Black
'Document security.
Dim security As PdfSecurity = document.Security
'Specifies key size and encryption algorithm using 128 bit key in RC4 mode.
security.KeySize = PdfEncryptionKeySize.Key128Bit
security.Algorithm = PdfEncryptionAlgorithm.RC4
security.OwnerPassword = "syncfusion"
'It allows printing and accessibility copy content
security.Permissions = PdfPermissionsFlags.Print Or
PdfPermissionsFlags.AccessibilityCopyContent
security.UserPassword = "password"
graphics.DrawString("This document is protected with owner password", font,
brush, New PointF(0, 40))
'Save and close the document.
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm.

```

```

security.KeySize = PdfEncryptionKeySize.Key128Bit;
security.Algorithm = PdfEncryptionAlgorithm.RC4;
security.OwnerPassword = "syncfusion";
//It allows printing and accessibility copy content
security.Permissions = PdfPermissionsFlags.Print |
PdfPermissionsFlags.AccessibilityCopyContent;
security.UserPassword = "password";
graphics.DrawString("This document is protected with owner password", font,
brush, new PointF(0, 40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm.
security.KeySize = PdfEncryptionKeySize.Key128Bit;
security.Algorithm = PdfEncryptionAlgorithm.RC4;
security.OwnerPassword = "syncfusion";
//It allows printing and accessibility copy content
security.Permissions = PdfPermissionsFlags.Print |
PdfPermissionsFlags.AccessibilityCopyContent;
security.UserPassword = "password";
graphics.DrawString("This document is protected with owner password", font,
brush, new PointF(0, 40));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.

```

```

PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm.
security.KeySize = PdfEncryptionKeySize.Key128Bit;
security.Algorithm = PdfEncryptionAlgorithm.RC4;
security.OwnerPassword = "syncfusion";
//It allows printing and accessibility copy content
security.Permissions = PdfPermissionsFlags.Print |
PdfPermissionsFlags.AccessibilityCopyContent;
security.UserPassword = "password";
graphics.DrawString("This document is protected with owner password", font,
brush, new PointF(0, 40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

Working with AES Encryption

You can encrypt PDF document using AES algorithm with 40bit or 128bit or 256bit key size. The following code snippet illustrates how to encrypt the PDF document with the [UserPassword](#).

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm.
security.KeySize = PdfEncryptionKeySize.Key256Bit;

```



```

security.Algorithm = PdfEncryptionAlgorithm.AES;
security.UserPassword = "password";
graphics.DrawString("Encrypted with AES 256bit", font, brush, new PointF(0,
40));
//Save and close the document.
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a new PDF document.
Dim document As New PdfDocument()
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
Dim font As New PdfStandardFont(PdfFontFamily.TimesRoman, 20.0F,
PdfFontStyle.Bold)
Dim brush As PdfBrush = PdfBrushes.Black
'Document security.
Dim security As PdfSecurity = document.Security
'Specifies key size and encryption algorithm using 256 bit key in AES mode.
security.KeySize = PdfEncryptionKeySize.Key256Bit
security.Algorithm = PdfEncryptionAlgorithm.AES
security.UserPassword = "password"
graphics.DrawString("Encrypted with AES 256bit", font, brush, New PointF(0,
40))
'Save and close the document.
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm.
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
security.UserPassword = "password";
graphics.DrawString("Encrypted with AES 256bit", font, brush, new PointF(0,
40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm.
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
security.UserPassword = "password";
graphics.DrawString("Encrypted with AES 256bit", font, brush, new PointF(0,
40));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm.
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
security.UserPassword = "password";
graphics.DrawString("Encrypted with AES 256bit", font, brush, new PointF(0,
40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}
```

You can protect the PDF document from printing, editing, copying with the [OwnerPassword](#) by using the following code snippet.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm using 256 bit key in AES mode.
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
security.OwnerPassword = "syncfusion";
//It allows printing and accessibility copy content
security.Permissions = PdfPermissionsFlags.Print |
PdfPermissionsFlags.AccessibilityCopyContent;
security.UserPassword = "password";
graphics.DrawString("This document is protected with owner password", font,
brush, new PointF(0, 40));
//Save and close the document.
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
Dim font As New PdfStandardFont(PdfFontFamily.TimesRoman, 20.0F,
PdfFontStyle.Bold)
Dim brush As PdfBrush = PdfBrushes.Black
'Document security.
Dim security As PdfSecurity = document.Security
'Specifies key size and encryption algorithm using 256 bit key in RC4 mode.
security.KeySize = PdfEncryptionKeySize.Key256Bit
```

```

security.Algorithm = PdfEncryptionAlgorithm.AES
security.OwnerPassword = "syncfusion"
'It allows printing and accessibility copy content
security.Permissions = PdfPermissionsFlags.Print Or
PdfPermissionsFlags.AccessibilityCopyContent
security.UserPassword = "password"
graphics.DrawString("This document is protected with owner password", font,
brush, New PointF(0, 40))
'Save and close the document.
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm using 256 bit key in AES mode.
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
security.OwnerPassword = "syncfusion";
//It allows printing and accessibility copy content
security.Permissions = PdfPermissionsFlags.Print |
PdfPermissionsFlags.AccessibilityCopyContent;
security.UserPassword = "password";
graphics.DrawString("This document is protected with owner password", font,
brush, new PointF(0, 40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm using 256 bit key in AES mode.
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;

```

```

security.OwnerPassword = "syncfusion";
//It allows printing and accessibility copy content
security.Permissions = PdfPermissionsFlags.Print |
PdfPermissionsFlags.AccessibilityCopyContent;
security.UserPassword = "password";
graphics.DrawString("This document is protected with owner password", font,
brush, new PointF(0, 40));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security.
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm using 256 bit key in AES mode.
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
security.OwnerPassword = "syncfusion";
//It allows printing and accessibility copy content
security.Permissions = PdfPermissionsFlags.Print |
PdfPermissionsFlags.AccessibilityCopyContent;
security.UserPassword = "password";
graphics.DrawString("This document is protected with owner password", font,
brush, new PointF(0, 40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}

```

```

}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

Encryption Options

Now, the Syncfusion PDF library has provided options to encrypt the PDF document as follows:

- Encrypt all contents** – All contents of the document will be encrypted. *Encrypt all contents except Metadata* – All contents of the document will be encrypted except metadata. **Encrypt only attachments** – Encrypts only the file attachments, rest of the document will be left unencrypted.

The default value of EncryptionOptions is EncryptAllContents. You can choose any one of these options using the property “EncryptionOptions” available in the class [PdfSecurity](#).

Encrypt all contents

You can encrypt all the PDF content by using the EncryptAllContents option available in the EncryptionOptions. The following code snippet explains how to encrypt all contents of the PDF document.

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptAllContents;
security.UserPassword = "password";
graphics.DrawString("Encrypted with AES 256bit", font, brush, new PointF(0,
40));
//Save and close the document
document.Save("Output.pdf");
document.Close();

```

VB.NET

```

'Create a new PDF document
Dim document As New PdfDocument()
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
Dim font As New PdfStandardFont(PdfFontFamily.TimesRoman, 20.0F,
PdfFontStyle.Bold)

```

```

Dim brush As PdfBrush = PdfBrushes.Black
'Document security
Dim security As PdfSecurity = document.Security
'Specifies key size and encryption algorithm using 256 bit key in AES mode
security.KeySize = PdfEncryptionKeySize.Key256Bit
security.Algorithm = PdfEncryptionAlgorithm.AES
'Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptAllContents;
security.UserPassword = "password"
graphics.DrawString("Encrypted with AES 256bit", font, brush, New PointF(0,
40))
'Save and close the document
document.Save("Output.pdf")
document.Close()

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptAllContents;
security.UserPassword = "password";
graphics.DrawString("Encrypted with AES 256bit", font, brush, new PointF(0,
40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key256Bit;

```

```

security.Algorithm = PdfEncryptionAlgorithm.AES;
//Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptAllContents;
security.UserPassword = "password";
graphics.DrawString("Encrypted with AES 256bit", font, brush, new PointF(0,
40));
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream); stream.Position = 0;
//Close the documents
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptAllContents;
security.UserPassword = "password";
graphics.DrawString("Encrypted with AES 256bit", font, brush, new PointF(0,
40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file //The operation in Save under Xamarin varies
between Windows Phone, Android, and iOS platforms. Refer to the PDF/Xamarin
section for respective code samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```



```
}

```

Encrypt all contents except metadata

The Syncfusion Essential PDF library now supports encrypting the PDF document except the document information (metadata) by using the `EncryptAllContentsExceptMetadata` option. The document information will not be encrypted when using this `EncryptionOption`.

The following code snippet explains how to encrypt all contents except metadata of the PDF document.

Note: Encrypt all contents except metadata is only supported in AES algorithms with 128bit, 256bit, and 256bit revision6 key size

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Specifies encryption option
security.EncryptionOptions =
PdfEncryptionOptions.EncryptAllContentsExceptMetadata;
security.UserPassword = "password";
graphics.DrawString("Encrypted all contents except metadata with AES
256bit", font, brush, new PointF(0, 40));
//Save and close the document
document.Save("Output.pdf");
document.Close();
```

VB.NET

```
'Create a new PDF document
Dim document As New PdfDocument()
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
Dim font As New PdfStandardFont(PdfFontFamily.TimesRoman, 20.0F,
PdfFontStyle.Bold)
Dim brush As PdfBrush = PdfBrushes.Black
'Document security
Dim security As PdfSecurity = document.Security
'Specifies key size and encryption algorithm using 256 bit key in AES mode
security.KeySize = PdfEncryptionKeySize.Key256Bit
security.Algorithm = PdfEncryptionAlgorithm.AES
'Specifies encryption option
security.EncryptionOptions =
PdfEncryptionOptions.EncryptAllContentsExceptMetadata
security.UserPassword = "password"
graphics.DrawString("Encrypted all contents except metadata with AES
256bit", font, brush, New PointF(0, 40))
```

```
'Save and close the document
document.Save("Output.pdf")
document.Close()
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Specifies encryption option
security.EncryptionOptions =
PdfEncryptionOptions.EncryptAllContentsExceptMetadata;
security.UserPassword = "password";
graphics.DrawString("Encrypted with all contents except metadata AES
256bit", font, brush, new PointF(0, 40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Specifies encryption option
security.EncryptionOptions =
PdfEncryptionOptions.EncryptAllContentsExceptMetadata;
security.UserPassword = "password";
graphics.DrawString("Encrypted all contents except metadata with AES
256bit", font, brush, new PointF(0, 40));
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream); stream.Position = 0;
```

```
//Close the documents
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Specifies encryption option
security.EncryptionOptions =
PdfEncryptionOptions.EncryptAllContentsExceptMetadata;
security.UserPassword = "password";
graphics.DrawString("Encrypted all contents except metadata with AES
256bit", font, brush, new PointF(0, 40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file //The operation in Save under Xamarin varies
between Windows Phone, Android, and iOS platforms. Refer to the PDF/Xamarin
section for respective code samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}
```

Encrypt only attachments

You can encrypt only attachments present in the PDF document by using the `EncryptOnlyAttachments` option available in the `EncryptionOptions`.

The following code example explains how to create an encrypt only attachment document using the Syncfusion PDF Library.

Note: [UserPassword](#) is mandatory for encrypt only attachments and it is only supported in AES algorithms with 128bit, 256bit, and 256bit revision6 key size

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptOnlyAttachments;
security.UserPassword = "password";
graphics.DrawString("Encrypted only attachments with AES 256bit", font,
brush, new PointF(0, 40));
//Creates an attachment
PdfAttachment attachment = new PdfAttachment("Input.txt");
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Add the attachment to the document
document.Attachments.Add(attachment);
//Save and close the document
document.Save("Output.pdf");
document.Close();
```

VB.NET

```
'Create a new PDF document
Dim document As New PdfDocument()
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
Dim font As New PdfStandardFont(PdfFontFamily.TimesRoman, 20.0F,
PdfFontStyle.Bold)
Dim brush As PdfBrush = PdfBrushes.Black
'Document security
Dim security As PdfSecurity = document.Security
'Specifies key size and encryption algorithm using 256 bit key in AES mode
security.KeySize = PdfEncryptionKeySize.Key256Bit
security.Algorithm = PdfEncryptionAlgorithm.AES
'Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptOnlyAttachments;
security.UserPassword = "password"
graphics.DrawString("Encrypted only attachments with AES 256bit", font,
brush, New PointF(0, 40))
'Creates an attachment
```

```

Dim attachment As New PdfAttachment("Input.txt")
attachment.ModificationDate = DateTime.Now
attachment.Description = "Input.txt"
attachment.MimeType = "application/txt"
Add the attachment to the document
document.Attachments.Add(attachment)
'Save and close the document
document.Save("Output.pdf")
document.Close()

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptOnlyAttachments;
security.UserPassword = "password";
graphics.DrawString("Encrypted only attachments with AES 256bit", font,
brush, new PointF(0, 40));
//Creates an attachment
PdfAttachment attachment = new PdfAttachment("Input.txt");
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Add the attachment to the document
document.Attachments.Add(attachment);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security
PdfSecurity security = document.Security;

```

```
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptOnlyAttachments;
security.UserPassword = "password";
graphics.DrawString("Encrypted only attachments with AES 256bit", font,
brush, new PointF(0, 40));
//Creates an attachment
PdfAttachment attachment = new PdfAttachment("Input.txt");
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Add the attachment to the document
document.Attachments.Add(attachment);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream); stream.Position = 0;
//Close the documents
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 20f,
PdfFontStyle.Bold);
PdfBrush brush = PdfBrushes.Black;
//Document security
PdfSecurity security = document.Security;
//Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptOnlyAttachments;
security.UserPassword = "password";
graphics.DrawString("Encrypted only attachments with AES 256bit", font,
brush, new PointF(0, 40));
//Creates an attachment
PdfAttachment attachment = new PdfAttachment("Input.txt");
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Add the attachment to the document
document.Attachments.Add(attachment);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
```

```
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file //The operation in Save under Xamarin varies
between Windows Phone, Android, and iOS platforms. Refer to the PDF/Xamarin
section for respective code samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}
```

Opening an encrypt-only-attachment document

The Syncfusion Essential PDF library now provides support for loading the encrypt-only-attachment PDF documents. To access the attachments in the existing PDF document, the [UserPassword](#) is mandatory.

You can provide the [UserPassword](#) in following ways:

Load the PDF document with password. Provide password using the OnPdfPassword Event when accessing the attachments.

It is possible to access all the contents except attachment when loading the PDF document without [UserPassword](#).

The following code example explains how to load an encrypt-only-attachment document with password using Syncfusion PDF Library.

C#

```
//Load the PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf", "password");
//Accessing the attachments
foreach (PdfAttachment attachment in document.Attachments)
{
    FileStream stream = new FileStream(attachment.FileName, FileMode.Create);
    stream.Write(attachment.Data, 0, attachment.Data.Length);
    stream.Dispose();
}
//Close the document
document.Close(true);
```

VB.NET

```
'Load the PDF document
Dim document As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf",
"password")
'Accessing the attachments
For Each attachment As PdfAttachment In document.Attachments
    Dim stream = New FileStream(attachment.FileName, FileMode.Create)
    stream.Write(attachment.Data, 0, attachment.Data.Length)
```

```
stream.Dispose
Next
'Close the document
document.Close(true)
```

UWP

```
//Load the PDF document as stream
Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument(pdfStream, "password");
//Accessing the attachments
foreach(PdfAttachment attachment in document.Attachments)
{
    FileStream stream= new FileStream(attachment.FileName, FileMode.Create);
    stream.Write(attachment.Data, 0, attachment.Data.Length);
    stream.Dispose();
}
//Close the document
document.Close(true);
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream, "password");
//Accessing the attachments
foreach(PdfAttachment attachment in document.Attachments)
{
    FileStream stream = new FileStream(attachment.FileName, FileMode.Create);
    stream.Write(attachment.Data, 0, attachment.Data.Length);
    stream.Dispose();
}
//Close the document
document.Close(true);
```

XAMARIN

```
//Load the PDF document as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument(docStream, "password");
//Accessing the attachments
foreach(PdfAttachment attachment in document.Attachments)
{
    FileStream stream= new FileStream(attachment.FileName, FileMode.Create);
    stream.Write(attachment.Data, 0, attachment.Data.Length);
    stream.Dispose();
}
//Close the document
```



```
document.Close(true);
```

Set user password using event when accessing the attachment

The following code example illustrates how to provide the password when accessing attachments from encrypt-only-attachment document using the OnPdfPassword event.

C#

```
//Load the PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
document.OnPdfPassword += LDoc_OnPdfPassword;
//Accessing the attachments
foreach(PdfAttachment attachment in document.Attachments)
{
    FileStream stream= new FileStream(attachment.FileName, FileMode.Create);
    stream.Write(attachment.Data, 0, attachment.Data.Length);
    stream.Dispose();
}
//Close the document
document.Close(true);
//Provide the user password in event
private static void LDoc_OnPdfPassword(object sender, OnPdfPasswordEventArgs
args)
{
    args.UserPassword = "syncfusion";
}
```

VB.NET

```
'Load the PDF document
Dim document As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'document.OnPdfPassword += LDoc_OnPdfPassword()
AddHandler document.OnPdfPassword, AddressOf LDoc_OnPdfPassword
'Accessing the attachments
For Each attachment As PdfAttachment In document.Attachments
    Dim stream As FileStream = New FileStream(attachment.FileName,
    FileMode.Create)
    stream.Write(attachment.Data, 0, attachment.Data.Length)
    stream.Dispose()
Next
'Close the document
document.Close(True)
'Provide the user password in event
Private Sub LDoc_OnPdfPassword(ByVal sender As Object, ByVal args As
OnPdfPasswordEventArgs)
    args.UserPassword = "password"
End Sub
```

UWP

```
//Load the PDF document as stream
Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.pdf");
//Creates an empty PDF loaded document instance
```

```

PdfLoadedDocument document = new PdfLoadedDocument(pdfStream);
document.OnPdfPassword += LDoc_OnPdfPassword;
//Accessing the attachments
foreach(PdfAttachment attachment in document.Attachments)
{
    FileStream stream = new FileStream(attachment.FileName, FileMode.Create);
    stream.Write(attachment.Data, 0, attachment.Data.Length);
    stream.Dispose();
}
//Close the document
document.Close(true);
//Provide the user password in event
private static void LDoc_OnPdfPassword(object sender, OnPdfPasswordEventArgs
args)
{
    args.UserPassword = "syncfusion";
}

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
document.OnPdfPassword += LDoc_OnPdfPassword;
//Accessing the attachments
foreach(PdfAttachment attachment in document.Attachments)
{
    FileStream stream= new FileStream(attachment.FileName, FileMode.Create);
    stream.Write(attachment.Data, 0, attachment.Data.Length);
    stream.Dispose();
}
//Close the document
document.Close(true);
//Provide the user password in event
private static void LDoc_OnPdfPassword(object sender, OnPdfPasswordEventArgs
args)
{
    args.UserPassword = "syncfusion";
}

```

XAMARIN

```

//Load the PDF document as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
document.OnPdfPassword += LDoc_OnPdfPassword;
// Accessing the attachments
foreach(PdfAttachment attachment in document.Attachments)
{
    FileStream stream= new FileStream(attachment.FileName, FileMode.Create);
    stream.Write(attachment.Data, 0, attachment.Data.Length);
    stream.Dispose();
}

```

```

}
//Close the document
document.Close(true);
//Provide the user password in event
private static void LDoc_OnPdfPassword(object sender, OnPdfPasswordEventArgs
args)
{
args.UserPassword = "syncfusion";
}

```

Protect attachments in existing PDF document

The Syncfusion PDF Library supports encrypting only the attachment files in an existing PDF document using the EncryptOnlyAttachments encryption option. Refer to the following code snippet.

Note: [UserPassword](#) is mandatory for this encryption option.

C#

```

//Load the PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//PDF document security
PdfSecurity security = document.Security;
//Specifies encryption key size, algorithm and permission
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Provide user password
security.UserPassword = "password";
//Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptOnlyAttachments;
//Save the document
document.Save("Output.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Load the PDF document
Dim document As New PdfLoadedDocument("Input.pdf")
'PDF document security
Dim security As PdfSecurity = document.Security
'Specifies encryption key size, algorithm and permission
security.KeySize = PdfEncryptionKeySize.Key256Bit
security.Algorithm = PdfEncryptionAlgorithm.AES
'Provide user password
security.UserPassword = "password"
'Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptOnlyAttachments
'Save the document
document.Save("Output.pdf")
'Close the document
document.Close(true)

```

UWP

```

//Load the PDF document as stream

```

```

Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument(pdfStream);
//PDF document security
PdfSecurity security = document.Security;
//Specifies encryption key size, algorithm and permission
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Provide user password
security.UserPassword = "password";
//Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptOnlyAttachments;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//PDF document security
PdfSecurity security = document.Security;
//Specifies encryption key size, algorithm and permission
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Provide user password
security.UserPassword = "password";
//Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptOnlyAttachments;
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream); stream.Position = 0;
//Close the documents
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the PDF document as stream

```

```

Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//PDF document security
PdfSecurity security = document.Security;
//Specifies encryption key size, algorithm and permission
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Provide user password
security.UserPassword = "password";
//Specifies encryption option
security.EncryptionOptions = PdfEncryptionOptions.EncryptOnlyAttachments;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file //The operation in Save under Xamarin varies
between Windows Phone, Android, and iOS platforms. Refer to the PDF/Xamarin
section for respective code samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

Protect an existing document

You can protect an existing PDF document with both [UserPassword](#) and [OwnerPassword](#) by using the following code snippet.

C#

```

//Load the PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//PDF document security
PdfSecurity security = document.Security;
//Specifies encryption key size, algorithm and permission.
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Provide owner and user password.
security.OwnerPassword = "ownerPassword256";
security.UserPassword = "userPassword256";
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);

```

VB.NET

```

'Load an existing document.
Dim document As New PdfLoadedDocument("Input.pdf")
'Document Security
Dim security As PdfSecurity = document.Security
'Specifies key size and encryption algorithm
security.KeySize = PdfEncryptionKeySize.Key128Bit
security.Algorithm = PdfEncryptionAlgorithm.RC4
'Provide owner and user password.
security.OwnerPassword = "ownerPassword256"
security.UserPassword = "userPassword256"
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)

```

UWP

```

//Load the PDF document as stream
Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument(pdfStream);
//PDF document security
PdfSecurity security = document.Security;
//Specifies encryption key size, algorithm and permission.
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Provide owner and user password.
security.OwnerPassword = "ownerPassword256";
security.UserPassword = "userPassword256";
MemoryStream memoryStream = new MemoryStream();
//Save the document.
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to pdf/uwp
section for respected code samples.
Save(memoryStream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//PDF document security
PdfSecurity security = document.Security;
//Specifies encryption key size, algorithm and permission.
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Provide owner and user password.
security.OwnerPassword = "ownerPassword256";
security.UserPassword = "userPassword256";

```

```
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//PDF document security
PdfSecurity security = document.Security;
//Specifies encryption key size, algorithm and permission.
security.KeySize = PdfEncryptionKeySize.Key256Bit;
security.Algorithm = PdfEncryptionAlgorithm.AES;
//Provide owner and user password.
security.OwnerPassword = "ownerPassword256";
security.UserPassword = "userPassword256";
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}
```

Changing the password of the PDF document

You can change the [UserPassword](#) of the existing PDF document by using following code snippet.

C#

```
//Load the password protected PDF document
```

```

PdfLoadedDocument loadedDocument = new
PdfLoadedDocument("Input.pdf", "password");
//Change the user password
loadedDocument.Security.UserPassword = "NewPassword";
//Save the password changed PDF document
loadedDocument.Save("Output.pdf");
loadedDocument.Close(true);

```

VB.NET

```

'Load the password protected PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf", "password")
'Change the user password
loadedDocument.Security.UserPassword = "NewPassword"
'Save the password changed PDF document
loadedDocument.Save("Output.pdf")
loadedDocument.Close(True)

```

UWP

```

//Load the PDF document as stream
Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new
PdfLoadedDocument(pdfStream, "password");
//Change the user password
loadedDocument.Security.UserPassword = "NewPassword";
MemoryStream memoryStream = new MemoryStream();
//Save the document.
loadedDocument.Save(memoryStream);
//Close the documents.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to pdf/uwp
section for respected code samples.
Save(memoryStream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream,
"password");
//Change the user password
loadedDocument.Security.UserPassword = "NewPassword";
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the documents.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";

```



```
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream,
"password");
//Change the user password
loadedDocument.Security.UserPassword = "NewPassword";
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Closes the document
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}
```

Change the permission of the PDF document

You can change the permission of the PDF document using the [Permissions](#). The following code snippet illustrates the same.

C#

```
//Load the password protected PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf",
"syncfusion");
//Change the permission
loadedDocument.Security.Permissions = PdfPermissionsFlags.CopyContent |
PdfPermissionsFlags.AssembleDocument;
//Save the PDF document
loadedDocument.Save("Output.pdf");
loadedDocument.Close(true);
```

VB.NET

```

'Load the password protected PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf", "syncfusion")
'Change the permission
loadedDocument.Security.Permissions = PdfPermissionsFlags.CopyContent Or
PdfPermissionsFlags.AssembleDocument
'Save the PDF document
loadedDocument.Save("Output.pdf")
loadedDocument.Close(True)

```

UWP

```

//Load the PDF document as stream
Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new
PdfLoadedDocument(pdfStream, "syncfusion");
//Change the permission
loadedDocument.Security.Permissions = PdfPermissionsFlags.CopyContent |
PdfPermissionsFlags.AssembleDocument;
MemoryStream memoryStream = new MemoryStream();
//Save the document.
loadedDocument.Save(memoryStream);
//Close the documents.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to pdf/uwp
section for respected code samples.
Save(memoryStream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream,
"syncfusion");
//Change the permission
loadedDocument.Security.Permissions = PdfPermissionsFlags.CopyContent |
PdfPermissionsFlags.AssembleDocument;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the documents.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
    Input.pdf");
PdfLoadedDocument loadedDocument = new
PdfLoadedDocument(docStream, "syncfusion");
//Change the permission
loadedDocument.Security.Permissions = PdfPermissionsFlags.CopyContent |
PdfPermissionsFlags.AssembleDocument;
//document.Attachments.RemoveAt(1);
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
loadedDocument.Save(memoryStream);
//Close the documents.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer pdf/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
    "application/pdf", memoryStream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
    "application/pdf", memoryStream);
}

```

Remove password from the user password PDF document

You can remove the [UserPassword](#) from the encrypted PDF document by using the following code snippet.

C#

```

//Load the password protected PDF document
PdfLoadedDocument loadedDocument = new
PdfLoadedDocument("Input.pdf", "password");
//Change the user password
loadedDocument.Security.UserPassword = string.Empty;
//Save the password removed PDF document
loadedDocument.Save("Output.pdf");
loadedDocument.Close(true);

```

VB.NET

```

'Load the password protected PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf", "password")
'Change the user password
loadedDocument.Security.UserPassword = String.Empty
'Save the password removed PDF document
loadedDocument.Save("Output.pdf")
loadedDocument.Close(True)

```

UWP

```
//Load the PDF document as stream
Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.Data.Input.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new
PdfLoadedDocument(pdfStream, "password");
//Change the user password
loadedDocument.Security.UserPassword = string.Empty;
MemoryStream memoryStream = new MemoryStream();
//Save the document.
loadedDocument.Save(memoryStream);
//Close the documents.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to pdf/uwp
section for respected code samples.
Save(memoryStream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream,
"password");
//Change the user password
loadedDocument.Security.UserPassword = string.Empty;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the documents.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new
PdfLoadedDocument(docStream, "password");
//Change the user password
loadedDocument.Security.UserPassword = string.Empty;
//Save the document into stream.
```

```

MemoryStream memoryStream = new MemoryStream();
loadedDocument.Save(memoryStream);
//Close the documents.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer pdf/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
        "application/pdf", memoryStream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
        "application/pdf", memoryStream);
}

```

How to determine whether the PDF document is password protected or not?

You can determine whether the existing PDF document is password protected or not by catching the [PdfDocumentException](#) as shown below.

C#

```

try
{
    //Load the password protected PDF document without user password
    PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Output.pdf");
}
catch (PdfDocumentException exception)
{
    if (exception.Message == "Can't open an encrypted document. The password is
        invalid.")
    {
        MessageBox.Show("Cannot open an encrypted document without password");
    }
}

```

VB.NET

```

Try
    'Load the password protected PDF document without user password
    Dim loadedDocument As New PdfLoadedDocument("Output.pdf")
    Catch exception As PdfDocumentException
    If exception.Message = "Can't open an encrypted document. The password is
        invalid." Then
        MessageBox.Show("Cannot open an encrypted document without password")
    End If
End Try

```

UWP

```

try

```

```

{
    //Load the PDF document as stream
    Stream pdfStream =
    typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
    sets.Data.Output.pdf");
    //Creates an empty PDF loaded document instance
    PdfLoadedDocument loadedDocument = new PdfLoadedDocument(pdfStream);
}
catch (PdfDocumentException exception)
{
}
}

```

ASP.NET CORE

```

try
{
    //Load the PDF document
    FileStream docStream = new FileStream("Output.pdf", FileMode.Open,
    FileAccess.Read);
    PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
}
catch (PdfDocumentException exception)
{
}
}

```

XAMARIN

```

try
{
    //Load the file as stream
    Stream docStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
    Output.pdf");
    PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
}
catch (PdfDocumentException exception)
{
}
}

```

How to determine whether the PDF document is protected by user or owner password
 Essential PDF supports identifying the document whether it is protected by user or owner.

The following table shows the various combination for loading the secured document with user or owner password:

Document type	Open with	User password	Owner password
PDF document secured with both the owner and user passwords.	User password	Returns user password	Returns null

Document type	Open with	User password	Owner password
PDF document secured with both the owner and user passwords.	Owner password	Returns user password Note: Returns null for AES 256 and AES 256 Revision 6 encryptions.	Returns owner password
PDF document secured with owner password alone.	Owner password	Returns null	Returns owner password
PDF document secured with user password alone.	User Password	Returns user password	Returns owner Password (owner password is same as the user password; it allows full permission to users).

Working with PDF Redaction

Removing sensitive content from the PDF document

Essential PDF supports removing or redacting the sensitive text and images from the PDF documents. Redaction is the process of permanently removing sensitive information from the PDF document, use the [PdfRedaction](#) class to remove content.

Note: Note: CJK text without TrueType font and complex script text cannot be redacted.

The following sample code snippet demonstrates the redaction of PDF documents from the specified bounds.

C#

```
//Load a PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
// Get first page from the document
PdfLoadedPage page = document.Pages[0] as PdfLoadedPage;
//Create PDF redaction for the page
PdfRedaction redaction = new PdfRedaction(new RectangleF(343, 147, 60, 17),
System.Drawing.Color.Black);
//Adds redaction to the loaded page
page.Redactions.Add(redaction);
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Load a PDF document
Dim document As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get first page from the document
Dim page As PdfLoadedPage = TryCast(document.Pages(0), PdfLoadedPage)
'Create PDF redaction for the page
Dim redaction As PdfRedaction = New PdfRedaction(New RectangleF(343, 147,
60, 17), System.Drawing.Color.Black)
'Adds redaction to the loaded page
page.Redactions.Add(redaction)
```

```
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

ASP.NET CORE

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

XAMARIN

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

Display text on the redacted area

You can draw overlay text on the redacted area using the [Appearance](#) property available in [PdfRedaction](#) class, and customize the overlay text with different font, style, color and brushes.

The following code snippet explains how to add overlay text in the redacted area.

C#

```
//Load a PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//Get first page from the document
PdfLoadedPage page = document.Pages[0] as PdfLoadedPage;
//Create PDF redaction for the page
PdfRedaction redaction = new PdfRedaction(new RectangleF(343, 147, 60, 17));
//Font for the overlay text
PdfStandardFont font = new PdfStandardFont(PdfFontFamily.Courier, 10);
//Draw text on the redacted area
redaction.Appearance.Graphics.DrawString("Redacted", font, PdfBrushes.Red,
new PointF(5, 5));
//Adds redaction to the loaded page
page.Redactions.Add(redaction);
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Load a PDF document
Dim document As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get first page from the document
Dim page As PdfLoadedPage = TryCast(document.Pages(0), PdfLoadedPage)
'Create PDF redaction for the page
Dim redaction As PdfRedaction = New PdfRedaction(New RectangleF(343, 147,
60, 17))
'Font for the overlay text
Dim font As PdfStandardFont = New PdfStandardFont(PdfFontFamily.Courier, 10)
'Draw text on the redacted area
```



```
redaction.Appearance.Graphics.DrawString("Redacted", font, PdfBrushes.Red,
New PointF(5, 5))
'Adds redaction to the loaded page
page.Redactions.Add(redaction)
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

ASP.NET CORE

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

XAMARIN

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

Drawing image on the redacted area

You can draw the image on the redacted area using the [Appearance](#) property in [PdfRedaction](#) class.

The following code snippet explains how to redact the information from a page by drawing image on the redacted area using appearance.

C#

```
//Load a PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//Get first page from the document
PdfLoadedPage page = document.Pages[0] as PdfLoadedPage;
//Create PDF redaction for the page
PdfRedaction redaction = new PdfRedaction(new RectangleF(63, 57, 182, 157));
//Draw image on the redacted bounds
PdfImage image = new PdfBitmap("Image.png");
redaction.Appearance.Graphics.DrawImage(image, new RectangleF(0, 0, 182,
157));
//Adds redaction to the loaded page
page.Redactions.Add(redaction);
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Load a PDF document
Dim document As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get first page from the document
Dim page As PdfLoadedPage = TryCast(document.Pages(0), PdfLoadedPage)
'Create PDF redaction for the page
Dim redaction As PdfRedaction = New PdfRedaction(New RectangleF(63, 57, 182,
157))
'Draw image on the redacted bounds
```

```

Dim image As PdfImage = New PdfBitmap("Image.png")
redaction.Appearance.Graphics.DrawImage(image, New RectangleF(0, 0, 182,
157))
'Draw image on the redacted bounds
page.Redactions.Add(redaction)
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

ASP.NET CORE

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

XAMARIN

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

Drawing pattern on the redacted area

You can draw the patterns on the redacted area using the [Appearance](#) property in [PdfRedaction](#) class.

The following code snippet explains how to redact the information from a page by drawing mosaic pattern on the redacted area using appearance.

C#

```

//Load a PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//Get first page from the document
PdfLoadedPage page = document.Pages[0] as PdfLoadedPage;
//Create PDF redaction for the page
PdfRedaction redaction = new PdfRedaction(new RectangleF(341, 149, 64, 14));
//Draw mosaic pattern on the redaction bounds
RectangleF rect = new RectangleF(0, 0, 8, 8);
PdfTilingBrush tilingBrush = new PdfTilingBrush(rect);
tilingBrush.Graphics.DrawRectangle(PdfBrushes.Gray, new RectangleF(0, 0, 2,
2));
tilingBrush.Graphics.DrawRectangle(PdfBrushes.White, new RectangleF(2, 0,
2, 2));
tilingBrush.Graphics.DrawRectangle(PdfBrushes.LightGray, new RectangleF(4,
0, 2, 2));
tilingBrush.Graphics.DrawRectangle(PdfBrushes.DarkGray, new RectangleF(6,
0, 2, 2));
tilingBrush.Graphics.DrawRectangle(PdfBrushes.White, new RectangleF(0, 2,
2, 2));
tilingBrush.Graphics.DrawRectangle(PdfBrushes.LightGray, new RectangleF(2,
2, 2, 2));
tilingBrush.Graphics.DrawRectangle(PdfBrushes.Black, new RectangleF(4, 2,
2, 2));
tilingBrush.Graphics.DrawRectangle(PdfBrushes.LightGray, new RectangleF(6,
2, 2, 2));

```

```

tillingBrush.Graphics.DrawRectangle(PdfBrushes.LightGray, new RectangleF(0,
4, 2, 2));
tillingBrush.Graphics.DrawRectangle(PdfBrushes.DarkGray, new RectangleF(2,
4, 2, 2));
tillingBrush.Graphics.DrawRectangle(PdfBrushes.LightGray, new RectangleF(4,
4, 2, 2));
tillingBrush.Graphics.DrawRectangle(PdfBrushes.White, new RectangleF(6, 4,
2, 2));
tillingBrush.Graphics.DrawRectangle(PdfBrushes.Black, new RectangleF(0, 6,
2, 2));
tillingBrush.Graphics.DrawRectangle(PdfBrushes.LightGray, new RectangleF(2,
6, 2, 2));
tillingBrush.Graphics.DrawRectangle(PdfBrushes.Black, new RectangleF(4, 6,
2, 2));
tillingBrush.Graphics.DrawRectangle(PdfBrushes.DarkGray, new RectangleF(6,
6, 2, 2));
rect = new RectangleF(0, 0, 16, 14);
PdfTilingBrush tillingBrushNew = new PdfTilingBrush(rect);
tillingBrushNew.Graphics.DrawRectangle(tillingBrush, rect);
redaction.Appearance.Graphics.DrawRectangle(tillingBrushNew, new
RectangleF(0, 0, 64, 14));
//Adds redaction to the loaded page
page.Redactions.Add(redaction);
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Load a PDF document
Dim document As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get first page from the document
Dim page As PdfLoadedPage = TryCast(document.Pages(0), PdfLoadedPage)
'Create PDF redaction for the page
Dim redaction As PdfRedaction = New PdfRedaction(New RectangleF(341, 149,
64, 14))
'Draw mosaic pattern on the redaction bounds
Dim rect As RectangleF = New RectangleF(0, 0, 8, 8)
Dim tillingBrush As PdfTilingBrush = New PdfTilingBrush(rect)
tillingBrush.Graphics.DrawRectangle(PdfBrushes.Gray, New RectangleF(0, 0, 2,
2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.White, New RectangleF(2, 0,
2, 2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.LightGray, New RectangleF(4,
0, 2, 2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.DarkGray, New RectangleF(6,
0, 2, 2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.White, New RectangleF(0, 2,
2, 2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.LightGray, New RectangleF(2,
2, 2, 2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.Black, New RectangleF(4, 2,
2, 2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.LightGray, New RectangleF(6,
2, 2, 2))

```

```

tillingBrush.Graphics.DrawRectangle(PdfBrushes.LightGray, New RectangleF(0,
4, 2, 2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.DarkGray, New RectangleF(2,
4, 2, 2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.LightGray, New RectangleF(4,
4, 2, 2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.White, New RectangleF(6, 4,
2, 2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.Black, New RectangleF(0, 6,
2, 2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.LightGray, New RectangleF(2,
6, 2, 2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.Black, New RectangleF(4, 6,
2, 2))
tillingBrush.Graphics.DrawRectangle(PdfBrushes.DarkGray, New RectangleF(6,
6, 2, 2))
rect = New RectangleF(0, 0, 16, 14)
Dim tillingBrushNew As PdfTilingBrush = New PdfTilingBrush(rect)
tillingBrushNew.Graphics.DrawRectangle(tillingBrush, rect)
redaction.Appearance.Graphics.DrawRectangle(tillingBrushNew, New
RectangleF(0, 0, 64, 14))
'Adds redaction to the loaded page
page.Redactions.Add(redaction)
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

ASP.NET CORE

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

XAMARIN

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

Fill color on the redacted area

You can draw the filled rectangle on the redacted bounds using the [FillColor](#) property in [PdfRedaction](#) class.

The following code snippet explains how to redact the information from a page with filled rectangle.

C#

```

//Load a PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//Get first page from the document
PdfLoadedPage page = document.Pages[0] as PdfLoadedPage;
//Create PDF redaction for the page
PdfRedaction redaction = new PdfRedaction(new RectangleF(343, 147, 60, 17));
//Set fill color for the redaction bounds
redaction.FillColor = System.Drawing.Color.Black;

```

```
//Adds redaction to the loaded page
page.Redactions.Add(redaction);
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Load a PDF document
Dim document As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get first page from the document
Dim page As PdfLoadedPage = TryCast(document.Pages(0), PdfLoadedPage)
'Create PDF redaction for the page
Dim redaction As PdfRedaction = New PdfRedaction(New RectangleF(343, 147,
60, 17))
'Set fill color for the redaction bounds
redaction.FillColor = System.Drawing.Color.Black
'Adds redaction to the loaded page
page.Redactions.Add(redaction)
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

ASP.NET CORE

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

XAMARIN

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

Redaction without fill color and appearance

You can redact PDF without drawing the filled rectangle or text on the redacted bounds.

The following code snippet explains how to redact the information from a page without drawing fill color and appearance on the redaction bounds.

C#

```
//Load a PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//Get first page from the document
PdfLoadedPage page = document.Pages[0] as PdfLoadedPage;
//Create PDF redaction for the page
PdfRedaction redaction = new PdfRedaction(new RectangleF(343, 147, 60, 17));
//Adds redaction to the loaded page
page.Redactions.Add(redaction);
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```

'Load a PDF document
Dim document As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get first page from the document
Dim page As PdfLoadedPage = TryCast(document.Pages(0), PdfLoadedPage)
'Create PDF redaction for the page
Dim redaction As PdfRedaction = New PdfRedaction(New RectangleF(343, 147,
60, 17))
'Adds redaction to the loaded page
page.Redactions.Add(redaction)
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

ASP.NET CORE

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

XAMARIN

```
//PDF supports redaction only in Windows Forms, WPF, ASP.NET, ASP.NET MVC.
```

Working with Digital Signature

Adding a digital signature

The Essential PDF allows you to add digital signature to the PDF document. To add digital signature, you need a certificate with private keys. The Essential PDF provides support for digital signature by using PFX files, Hardware Security Module(HSM), Online Certificate Status Protocol (OCSP), Certificate Revocation List (CRL), and Windows Certificate Store.

The following code example explains how to add a digital signature to the PDF document.

C#

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
PdfCertificate pdfCert = new PdfCertificate(@"PDF.pfx", "syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
PdfBitmap signatureImage = new PdfBitmap(@"signature.jpg");
//Sets signature information

```

```
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Saves and closes the document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Adds a new page
Dim page As PdfPageBase = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
'Creates a certificate instance from PFX file with private key
Dim pdfCert As New PdfCertificate("PDF.pfx", "syncfusion")
'Creates a digital signature
Dim signature As New PdfSignature(document, page, pdfCert, "Signature")
'Sets an image for signature field
Dim signatureImage As New PdfBitmap("signature.jpg")
'Sets signature info
signature.Bounds = New RectangleF(New PointF(0, 0),
signatureImage.PhysicalDimension)
signature.ContactInfo = "johndoe@owned.us"
signature.LocationInfo = "Honolulu, Hawaii"
signature.Reason = "I am author of this document."
'Draws the signature image
graphics.DrawImage(signatureImage, 0, 0)
'Saves and closes the document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.PDF.pfx");
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.signature.jpg");
```

```

PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
FileStream certificateStream = new FileStream("PDF.pfx", FileMode.Open,
FileAccess.Read);
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
FileStream imageStream = new FileStream("signature.jpg", FileMode.Open,
FileAccess.Read);
//Sets an image for signature field
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";

```



```
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF.pfx");
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.signature.jpg");
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Adding a digital signature using stream

The following code example illustrates how to add a digital signature in the PDF document using stream as follows.

C#

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Gets a stream from .pfx file
Stream pfxStream = File.OpenRead("PDF.pfx");
//Creates a certificate instance from PFX file stream with private key
PdfCertificate pdfCert = new PdfCertificate(pfxStream, "syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
PdfBitmap signatureImage = new PdfBitmap(@"signature.jpg");
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Saves and closes the document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Adds a new page
Dim page As PdfPageBase = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
'Gets stream from .pfx file
Dim pfxStream As Stream = File.OpenRead("PDF.pfx")
'Creates a certificate instance from PFX file stream with private key
Dim pdfCert As New PdfCertificate(pfxStream, "syncfusion")
'Creates a digital signature
Dim signature As New PdfSignature(document, page, pdfCert, "Signature")
'Sets an image for signature field
Dim signatureImage As New PdfBitmap("signature.jpg")
'Sets signature information
signature.Bounds = New RectangleF(New PointF(0, 0),
signatureImage.PhysicalDimension)
signature.ContactInfo = "johndoe@owned.us"
signature.LocationInfo = "Honolulu, Hawaii"
signature.Reason = "I am author of this document."
'Draws the signature image
graphics.DrawImage(signatureImage, 0, 0)
'Saves and closes the document
```

```
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.signature.jpg");
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
FileStream certificateStream = new FileStream("PDF.pfx", FileMode.Open,
FileAccess.Read);
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
```

```
//Sets an image for signature field
FileStream imageStream = new FileStream("signature.jpg", FileMode.Open,
FileAccess.Read);
//Sets an image for signature field
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF.pfx");
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.signature.jpg");
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
```

```

//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

You can add a digital signature to an existing document as follows.

C#

```

//Loads the PDF document with signature field
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Gets the page
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Creates a signature field
PdfSignatureField signatureField = new PdfSignatureField(page,
"SignatureField");
signatureField.Bounds = new RectangleF(0, 0, 100, 100);
signatureField.Signature = new PdfSignature();
//Adds certificate to the signature field
signatureField.Signature.Certificate = new PdfCertificate(@"PDF.pfx",
"syncfusion");
signatureField.Signature.Reason = "I am author of this document";
//Adds the field
loadedDocument.Form.Fields.Add(signatureField);
//Saves the certified PDF document
loadedDocument.Save(@"Output.pdf");
loadedDocument.Close(true);

```

VB.NET

```

'Loads the PDF document with signature field
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Gets the page
Dim page As PdfLoadedPage = TryCast(loadedDocument.Pages(0), PdfLoadedPage)
'Creates a signature field
Dim signatureField As New PdfSignatureField(page, "SignatureField")
signatureField.Bounds = New RectangleF(0, 0, 100, 100)
signatureField.Signature = New PdfSignature()
'Adds certificate to the signature field

```

```
signatureField.Signature.Certificate = New PdfCertificate("PDF.pfx",
"syncfusion")
signatureField.Signature.Reason = "I am author of this document"
'Adds the field
loadedDocument.Form.Fields.Add(signatureField)
'Saves the certified PDF document
loadedDocument.Save("Output.pdf")
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Gets the page
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Creates a signature field
PdfSignatureField signatureField = new PdfSignatureField(page,
"SignatureField");
signatureField.Bounds = new RectangleF(0, 0, 100, 100);
signatureField.Signature = new PdfSignature();
//Adds certificate to the signature field
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.
signatureField.Signature.Certificate = new PdfCertificate(certificateStream,
"syncfusion");
signatureField.Signature.Reason = "I am author of this document";
//Adds the field
loadedDocument.Form.Fields.Add(signatureField);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Gets the page
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Creates a signature field
```

```

PdfSignatureField signatureField = new PdfSignatureField(page,
"SignatureField");
signatureField.Bounds = new RectangleF(0, 0, 100, 100);
signatureField.Signature = new PdfSignature();
//Adds certificate to the signature field
FileStream certificateStream = new FileStream("PDF.pfx", FileMode.Open,
FileAccess.Read);
signatureField.Signature.Certificate = new PdfCertificate(certificateStream,
"syncfusion");
signatureField.Signature.Reason = "I am author of this document";
//Adds the field
loadedDocument.Form.Fields.Add(signatureField);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the documents
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Gets the page
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Creates a signature field
PdfSignatureField signatureField = new PdfSignatureField(page,
"SignatureField");
signatureField.Bounds = new RectangleF(0, 0, 100, 100);
signatureField.Signature = new PdfSignature();
//Adds certificate to the signature field
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF.pfx");
signatureField.Signature.Certificate = new PdfCertificate(certificateStream,
"syncfusion");
signatureField.Signature.Reason = "I am author of this document";
//Adds the field
loadedDocument.Form.Fields.Add(signatureField);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Closes the document
loadedDocument.Close(true);
//Save the stream into PDF file

```

```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Adding a digital signature using X509Certificate2

The following code example illustrates how to add digital signature in a PDF document using X509Certificate2 as follows.

C#

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
X509Certificate2 certificate = new X509Certificate2("PDF.pfx",
"syncfusion");
PdfCertificate pdfCertificate = new PdfCertificate(certificate);
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCertificate,
"Signature");
//Sets an image for signature field
PdfBitmap signatureImage = new PdfBitmap(@"signature.jpg");
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the document
document.Save("Output.pdf");
//Close the document
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Adds a new page
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
```



```

'Creates a certificate instance from PFX file with private key
Dim certificate As New X509Certificate2("PDF.pfx", "syncfusion")
Dim pdfCertificate As New PdfCertificate(certificate)
'Creates a digital signature
Dim signature As New PdfSignature(document, page, pdfCertificate,
"Signature")
'Sets an image for signature field
Dim signatureImage As New PdfBitmap("signature.jpg")
'Sets signature information
signature.Bounds = New RectangleF(New PointF(0, 0),
signatureImage.PhysicalDimension)
signature.ContactInfo = "johndoe@owned.us"
signature.LocationInfo = "Honolulu, Hawaii"
signature.Reason = "I am author of this document."
'Draws the signature image
graphics.DrawImage(signatureImage, 0, 0)
'Save the document
document.Save("Output.pdf")
'Close the document
document.Close(True)

```

UWP

```

//Essential PDF supports adding a digital signature using X509Certificate2
only in Windows Forms, WPF, ASP.NET, ASP.NET MVC, and ASP.NET Core
platforms.

```

ASP.NET CORE

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
X509Certificate2 certificate = new X509Certificate2("PDF.pfx",
"syncfusion");
PdfCertificate pdfCertificate = new PdfCertificate(certificate);
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCertificate,
"Signature");
//Sets an image for signature field
FileStream imageStream = new FileStream("signature.jpg", FileMode.Open,
FileAccess.Read);
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream

```

```
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF.pfx");
MemoryStream ms = new MemoryStream();
certificateStream.CopyTo(ms);
X509Certificate2 certificate = new X509Certificate2(ms.ToArray(),
"syncfusion");
PdfCertificate pdfCertificate = new PdfCertificate(certificate);
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCertificate,
"Signature");
//Sets an image for signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.signature.jpg");
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the document as stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Signing an existing document

You can load the signature field from the existing PDF document and add certificate to the document as follows.

C#

```
//Loads a PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Gets the first page of the document
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Gets the first signature field of the PDF document
PdfLoadedSignatureField field = loadedDocument.Form.Fields[0] as
PdfLoadedSignatureField;
//Creates a certificate
PdfCertificate certificate = new PdfCertificate(@"PDF.pfx", "syncfusion");
field.Signature = new
PdfSignature(loadedDocument, page, certificate, "Signature", field);
//Saves the document
loadedDocument.Save("Output.pdf");
//Closes the document
loadedDocument.Close(true);
```

VB.NET

```
'Loads a PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Gets the first page of the document
Dim page As PdfLoadedPage = TryCast(loadedDocument.Pages(0), PdfLoadedPage)
'Gets the first signature field of the PDF document
Dim field As PdfLoadedSignatureField =
TryCast(loadedDocument.Form.Fields(0), PdfLoadedSignatureField)
'Creates a certificate
Dim certificate As New PdfCertificate("PDF.pfx", "syncfusion")
field.Signature = New PdfSignature(loadedDocument, page, certificate,
"Signature", field)
'Saves the document
loadedDocument.Save("Output.pdf")
'Closes the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
```

```

StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Gets the first page of the document
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Gets the first signature field of the PDF document
PdfLoadedSignatureField field = loadedDocument.Form.Fields[0] as
PdfLoadedSignatureField;
//Creates a certificate
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.
PdfCertificate certificate = new PdfCertificate(certificateStream,
"syncfusion");
field.Signature = new PdfSignature(loadedDocument, page, certificate,
"Signature", field);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Gets the first page of the document
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Gets the first signature field of the PDF document
PdfLoadedSignatureField field = loadedDocument.Form.Fields[0] as
PdfLoadedSignatureField;
//Creates a certificate
FileStream certificateStream = new FileStream("PDF.pfx", FileMode.Open,
FileAccess.Read);
PdfCertificate certificate = new PdfCertificate(certificateStream,
"syncfusion");
field.Signature = new PdfSignature(loadedDocument, page, certificate,
"Signature", field);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the documents
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";

```

```
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Gets the first page of the document
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Gets the first signature field of the PDF document
PdfLoadedSignatureField field = loadedDocument.Form.Fields[0] as
PdfLoadedSignatureField;
//Creates a certificate
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF.pfx");
PdfCertificate certificate = new PdfCertificate(certificateStream,
"syncfusion");
field.Signature = new PdfSignature(loadedDocument, page, certificate,
"Signature", field);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Closes the document
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Sign an existing document using stream

You can load the signature field from an existing PDF document and add certificate to the document using stream as follows.

C#

```
//Loads a PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Gets the first page of the document
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
```

```

//Gets the first signature field of the PDF document
PdfLoadedSignatureField field = loadedDocument.Form.Fields[0] as
PdfLoadedSignatureField;
//Gets the stream from .pfx file
Stream pfxStream = File.OpenRead("PDF.pfx");
//Creates a certificate instance from PFX file stream with private key
PdfCertificate certificate = new PdfCertificate(pfxStream, "syncfusion");
field.Signature = new PdfSignature(loadedDocument, page, certificate,
"Signature", field);
//Saves the document
loadedDocument.Save("Output.pdf");
//Closes the document
loadedDocument.Close(true);

```

VB.NET

```

'Loads a PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Gets the first page of the document
Dim page As PdfLoadedPage = TryCast(loadedDocument.Pages(0), PdfLoadedPage)
'Gets the first signature field of the PDF document
Dim field As PdfLoadedSignatureField =
TryCast(loadedDocument.Form.Fields(0), PdfLoadedSignatureField)
'Gets the stream from .pfx file
Dim pfxStream As Stream = File.OpenRead("PDF.pfx")
'Creates a certificate instance from PFX file stream with private key
Dim certificate As New PdfCertificate(pfxStream, "syncfusion")
field.Signature = New PdfSignature(loadedDocument, page, certificate,
"Signature", field)
'Saves the document
loadedDocument.Save("Output.pdf")
'Closes the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Gets the first page of the document
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Gets the first signature field of the PDF document
PdfLoadedSignatureField field = loadedDocument.Form.Fields[0] as
PdfLoadedSignatureField;
//Creates a certificate
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.PDF.pfx");

```

```

PdfCertificate certificate = new PdfCertificate(certificateStream,
"syncfusion");
field.Signature = new PdfSignature(loadedDocument, page, certificate,
"Signature", field);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Gets the first page of the document
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Gets the first signature field of the PDF document
PdfLoadedSignatureField field = loadedDocument.Form.Fields[0] as
PdfLoadedSignatureField;
//Creates a certificate
FileStream certificateStream = new FileStream("PDF.pfx", FileMode.Open,
FileAccess.Read);
PdfCertificate certificate = new PdfCertificate(certificateStream,
"syncfusion");
field.Signature = new PdfSignature(loadedDocument, page, certificate,
"Signature", field);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the documents
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Gets the first page of the document
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Gets the first signature field of the PDF document

```

```

PdfLoadedSignatureField field = loadedDocument.Form.Fields[0] as
PdfLoadedSignatureField;
//Creates a certificate
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF.pfx");
PdfCertificate certificate = new PdfCertificate(certificateStream,
"syncfusion");
field.Signature = new PdfSignature(loadedDocument, page, certificate,
"Signature", field);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Closes the document
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Externally sing a PDF document

You can sign the PDF document from external digital signature created from various sources such as HSM, USB token, smart card, or other cloud services such as DigitalSign.

The following code example shows how to sign the PDF document from external signature.

C#

```

//Load existing PDF document
PdfLoadedDocument document = new PdfLoadedDocument("PDF_Succinctly.pdf");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, document.Pages[0], null,
"DigitalSignature");
signature.ComputeHash += Signature_ComputeHash;
//Save the PDF document
document.Save("ExternalSignature.pdf");
//Close the document
document.Close(true);
void Signature_ComputeHash(object sender, PdfSignatureEventArgs arguments)
{
//Get the document bytes
byte[] documentBytes = arguments.Data;
SignedCms signedCms = new SignedCms(new ContentInfo(documentBytes),
detached: true);
//Compute the signature using the specified digital ID file and the password

```



```

X509Certificate2 certificate = new
X509Certificate2("DigitalSignatureTest.pfx", "DigitalPass123");
var cmsSigner = new CmsSigner(certificate);
//Set the digest algorithm SHA256
cmsSigner.DigestAlgorithm = new Oid("2.16.840.1.101.3.4.2.1");
signedCms.ComputeSignature(cmsSigner);
//Embed the encoded digital signature to the PDF document
arguments.SignedData = signedCms.Encode();
}

```

VB.NET

```

'Load existing PDF document
Dim document As PdfLoadedDocument = New
PdfLoadedDocument("PDF_Succinctly.pdf")
'Creates a digital signature
Dim signature As PdfSignature = New PdfSignature(document,
document.Pages(0), Nothing, "DigitalSignature")
signature.ComputeHash += Signature_ComputeHash
'Save the PDF document
document.Save("ExternalSignature.pdf")
'Close the document
document.Close(True)
Private Sub Signature_ComputeHash(ByVal sender As Object, ByVal arguments As
PdfSignatureEventArgs)
'Get the document bytes
Dim documentBytes As Byte() = arguments.Data
Dim signedCms As SignedCms = New SignedCms(New ContentInfo(documentBytes),
detached:=True)
'Compute the signature using the specified digital ID file and the password
Dim certificate As X509Certificate2 = New
X509Certificate2("DigitalSignatureTest.pfx", "DigitalPass123")
Dim cmsSigner = New CmsSigner(certificate)
'Set the digest algorithm SHA256
cmsSigner.DigestAlgorithm = New Oid("2.16.840.1.101.3.4.2.1")
signedCms.ComputeSignature(cmsSigner)
'Embed the encoded digital signature to the PDF document
arguments.SignedData = signedCms.Encode()
End Sub

```

UWP

```

//Get the stream from the document
Stream documentStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.PDF_Succinctly.pdf");
//Load an existing signed PDF document
PdfLoadedDocument document = new PdfLoadedDocument(documentStream);
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, document.Pages[0], null,
"DigitalSignature");
signature.ComputeHash += Signature_ComputeHash;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document

```

```

document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, " ExternalSignature.pdf");
private static void Signature_ComputeHash1(object sender,
PdfSignatureEventArgs ars)
{
    //Get the document bytes
    byte[] documentBytes = ars.Data;
    SignedCms signedCms = new SignedCms(new ContentInfo(documentBytes),
detached: true);
    //Compute the signature using the specified digital ID file and the password
    X509Certificate2 certificate = new
X509Certificate2("DigitalSignatureTest.pfx", "DigitalPass123");
    var cmsSigner = new CmsSigner(certificate);
    //Set the digest algorithm SHA256
    cmsSigner.DigestAlgorithm = new Oid("2.16.840.1.101.3.4.2.1");
    signedCms.ComputeSignature(cmsSigner);
    //Embed the encoded digital signature to the PDF document
    ars.SignedData = signedCms.Encode();
}

```

ASP.NET CORE

```

//Get the stream from the document
FileStream documentStream = new FileStream("PDF_Succinctly.pdf ",
FileMode.Open, FileAccess.Read);
//Load the existing PDF document
PdfLoadedDocument document = new PdfLoadedDocument(documentStream);
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, document.Pages[0], null,
"DigitalSignature");
signature.ComputeHash += Signature_ComputeHash;
MemoryStream ms = new MemoryStream();
document.Save(ms);
document.Close(true);
private static void Signature_ComputeHash1(object sender,
PdfSignatureEventArgs ars)
{
    //Get the document bytes
    byte[] documentBytes = ars.Data;
    SignedCms signedCms = new SignedCms(new ContentInfo(documentBytes),
detached: true);
    //Compute the signature using the specified digital ID file and the password
    X509Certificate2 certificate = new
X509Certificate2("DigitalSignatureTest.pfx", "DigitalPass123");
    var cmsSigner = new CmsSigner(certificate);
    //Set the digest algorithm SHA256
    cmsSigner.DigestAlgorithm = new Oid("2.16.840.1.101.3.4.2.1");
    signedCms.ComputeSignature(cmsSigner);
    //Embed the encoded digital signature to the PDF document
    ars.SignedData = signedCms.Encode();
}

```

XAMARIN

```

//Get the stream from the document
Stream documentStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF_Succinctly.pdf");
//Load an existing signed PDF document
PdfLoadedDocument document = new PdfLoadedDocument(documentStream);
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, document.Pages[0], null,
"DigitalSignature");
signature.ComputeHash += Signature_ComputeHash;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("ExternalSigna
ture.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("ExternalSignature.pdf",
"application/pdf", stream);
}
private static void Signature_ComputeHash1(object sender,
PdfSignatureEventArgs ars)
{
//Get the document bytes
byte[] documentBytes = ars.Data;
SignedCms signedCms = new SignedCms(new ContentInfo(documentBytes),
detached: true);
//Compute the signature using the specified digital ID file and the password
X509Certificate2 certificate = new
X509Certificate2("DigitalSignatureTest.pfx", "DigitalPass123");
var cmsSigner = new CmsSigner(certificate);
//Set the digest algorithm SHA256
cmsSigner.DigestAlgorithm = new Oid("2.16.840.1.101.3.4.2.1");
signedCms.ComputeSignature(cmsSigner);
//Embed the encoded digital signature to the PDF document
ars.SignedData = signedCms.Encode();
}

```

Create Long Term Validation (LTV) when signing PDF documents externally

You can create a Long Term validation (LTV) when signing PDF documents externally using your private/public certificates.

The following code example shows how to create an LTV when signing a PDF document from external signature.

C#

```

//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get the page of the existing PDF document
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a new PDF signature without PdfCertificate instance
PdfSignature signature = new PdfSignature(loadedDocument, loadedPage, null,
"Signature1");
//Hook up the ComputeHash event
signature.ComputeHash += Signature_ComputeHash;
//Create X509Certificate2 from your certificate to create a long-term validity
X509Certificate2 x509 = new X509Certificate2("PDF.pfx", "password123");
//Create LTV with your public certificates
signature.CreateLongTermValidity(new List<X509Certificate2> { x509 });
//Save and close the PDF document
loadedDocument.Save("SignedDocument.pdf");
loadedDocument.Close(true);

```

VB.NET

```

'Load an existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the page of the existing PDF document
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Create a new PDF signature without PdfCertificate instance
Dim signature As PdfSignature = New PdfSignature(loadedDocument, loadedPage,
Nothing, "Signature1")
' Hook up the ComputeHash event
AddHandler signature.ComputeHash, AddressOf Signature_ComputeHash
'Create X509Certificate2 from your certificate to create a long-term validity
Dim x509 As X509Certificate2 = New X509Certificate2("PDF.pfx",
"password123")
'Create LTV with your public certificates
signature.CreateLongTermValidity(New List(Of X509Certificate2) From { x509
})
'Save and close the PDF document
loadedDocument.Save("SignedDocument.pdf")
loadedDocument.Close(True)

```

UWP

```

//Get the stream from the document
Stream documentStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(documentStream);
//Get the page of the existing PDF document
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a new PDF signature without PdfCertificate instance
PdfSignature signature = new PdfSignature(loadedDocument, loadedPage, null,
"Signature1");
//Hook up the ComputeHash event
signature.ComputeHash += Signature_ComputeHash;

```

```
//Create X509Certificate2 from your certificate to create a long-term
validity
X509Certificate2 x509 = new X509Certificate2("PDF.pfx", "password123");
//Create LTV with your public certificates
signature.CreateLongTermValidity(new List<X509Certificate2> { x509 });
//Save the PDF document
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Get the stream from the document
FileStream documentStream = new FileStream("Input.pdf ", FileMode.Open,
FileAccess.Read);
//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(documentStream);
//Get the page of the existing PDF document
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Create a new PDF signature without PdfCertificate instance
PdfSignature signature = new PdfSignature(loadedDocument, loadedPage, null,
"Signature1");
//Hook up the ComputeHash event
signature.ComputeHash += Signature_ComputeHash;
//Create X509Certificate2 from your certificate to create a long-term
validity
X509Certificate2 x509 = new X509Certificate2("PDF.pfx", "password123");
//Create LTV with your public certificates
signature.CreateLongTermValidity(new List<X509Certificate2> { x509 });
//Save the PDF document
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a File object by using the file contents, content type, and file
name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Get the stream from the document
Stream documentStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets. Input.pdf");
//Load an existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(documentStream);
//Get the page of the existing PDF document
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
```

```

//Create a new PDF signature without PdfCertificate instance
PdfSignature signature = new PdfSignature(loadedDocument, loadedPage, null,
"Signature1");
//Hook up the ComputeHash event
signature.ComputeHash += Signature_ComputeHash;
//Create X509Certificate2 from your certificate to create a long-term validity
X509Certificate2 x509 = new X509Certificate2("PDF.pfx", "password123");
//Create LTV with your public certificates
signature.CreateLongTermValidity(new List<X509Certificate2> { x509 });
//Save the PDF document
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android, and iOS platforms. Refer to the PDF/Xamarin section for respective code samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf", "application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf", "application/pdf", stream);
}

```

Adding a signature validation appearance based on the signature

You can add the dynamic signature validation appearance to the signature field by enabling the “EnableValidationAppearance” property available in the “PdfSignature” class, the appearance will change based on the PDF reader validation.

Refer to the following code sample.

C#

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Add a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics=page.Graphics;
//Creates a certificate instance from the PFX file with private key
PdfCertificate pdfCert = new PdfCertificate(@"PDF.pfx", "syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert, "Signature");
//Sets an image for the signature field
PdfBitmap signatureImage = new PdfBitmap(@"signature.jpg");
//Sets enable signature validation appearance
signature.EnableValidationAppearance = true;
//Sets signature information
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";

```

```
signature.Reason = "I am author of this document.";
//Draw the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save and close documents
document.Save("Output.pdf");
document.Close( true );
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Adds a new page
Dim page As PdfPageBase = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
'Creates a certificate instance from the PFX file with private key
Dim pdfCert As New PdfCertificate("PDF.pfx", "syncfusion")
'Creates a digital signature
Dim signature As New PdfSignature(document, page, pdfCert, "Signature")
'Sets an image for the signature field
Dim signatureImage As New PdfBitmap("signature.jpg")
'Sets enable signature validation appearance
signature.EnableValidationAppearance = True
'Sets signature info
signature.Bounds = New RectangleF(New PointF(0, 0),
signatureImage.PhysicalDimension)
signature.ContactInfo = "johndoe@owned.us"
signature.LocationInfo = "Honolulu, Hawaii"
signature.Reason = "I am author of this document."
'Draws the signature image
graphics.DrawImage(signatureImage, 0, 0)
'Saves and closes the document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from the PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.PDF.pfx");
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.signature.jpg");
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets enable signature validation appearance
```

```
signature.EnableValidationAppearance = true;
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from the PFX file with private key
FileStream certificateStream = new FileStream("PDF.pfx", FileMode.Open,
FileAccess.Read);
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for the signature field
FileStream imageStream = new FileStream("signature.jpg", FileMode.Open,
FileAccess.Read);
//Sets an image for signature field
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets enable signature validation appearance
signature.EnableValidationAppearance = true;
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
```



```
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

XAMARIN

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from the PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF.pfx");
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for the signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.signature.jpg");
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets enable signature validation appearance
signature.EnableValidationAppearance = true;
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
```

```
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Adding a timestamp in digital signature

Essential PDF allows you to add timestamp in the digital signature of the PDF document. The following code example explains the same.

C#

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
PdfCertificate pdfCert = new PdfCertificate(@"PDF.pfx", "syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(page, pdfCert, "Signature");
//Sets an image for signature field
PdfBitmap image = new PdfBitmap(@"syncfusion_logo.gif");
//Adds time stamp by using the server URI and credentials
signature.TimeStampServer = new TimeStampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
//Sets signature info
signature.Bounds = new RectangleF(new PointF(0, 0),
image.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(image, 0, 0);
//Saves and closes the document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Adds a new page
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
'Creates a certificate instance from PFX file with private key
Dim pdfCert As New PdfCertificate("PDF.pfx", "syncfusion")
'Creates a digital signature
Dim signature As New PdfSignature(page, pdfCert, "Signature")
'Sets an image for signature field
Dim image As New PdfBitmap("syncfusion_logo.gif")
```

```

'Adds time stamp by using the server URI and credentials
signature.TimeStampServer = New TimeStampServer(New
Uri("http://syncfusion.digistamp.com"), "user", "123456")
'Sets signature info
signature.Bounds = New RectangleF(New PointF(0, 0), image.PhysicalDimension)
signature.ContactInfo = "johndoe@owned.us"
signature.LocationInfo = "Honolulu, Hawaii"
signature.Reason = "I am author of this document."
'Draws the signature image
graphics.DrawImage(image, 0, 0)
'Saves and closes the document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.PDF.pfx");
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.syncfusion_logo.gif");
PdfBitmap image = new PdfBitmap(imageStream);
//Adds time stamp by using the server URI and credentials
signature.TimeStampServer = new TimeStampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
//Sets signature info
signature.Bounds = new RectangleF(new PointF(0, 0),
image.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(image, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
FileStream certificateStream = new FileStream("PDF.pfx", FileMode.Open,
FileAccess.Read);
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
FileStream imageStream = new FileStream("syncfusion_logo.gif",
FileMode.Open, FileAccess.Read);
//Sets an image for signature field
PdfBitmap image = new PdfBitmap(imageStream);
//Adds time stamp by using the server URI and credentials
signature.TimestampServer = new TimestampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
//Sets signature info
signature.Bounds = new RectangleF(new PointF(0, 0),
image.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(image, 0, 0);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF.pfx");
```

```

PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.syncfusion_logo.gif");
PdfBitmap image = new PdfBitmap(imageStream);
//Adds time stamp by using the server URI and credentials
signature.TimeStampServer = new TimeStampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
//Sets signature info
signature.Bounds = new RectangleF(new PointF(0, 0),
image.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(image, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("sample.pdf",
"application/pdf", stream);
}

```

Adding a timestamp to PDF document

You can add timestamp to the PDF document using the following code snippet.

C#

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPage page = document.Pages.Add();
//Creates a digital signature
PdfSignature signature = new PdfSignature(page, "Signature");
//Add the time stamp by using the server URI
signature.TimeStampServer = new TimeStampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");

```

```
//Save and close the document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Adds a new page
Dim page As PdfPage = document.Pages.Add()
'Creates a digital signature
Dim signature As New PdfSignature(page, "Signature")
'Adds time stamp by using the server URI
signature.TimestampServer = New TimestampServer(New
Uri("http://syncfusion.digistamp.com"), "user", "123456")
'Saves and closes the document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a new document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPage page = document.Pages.Add();
//Creates a digital signature
PdfSignature signature = new PdfSignature(page, "Signature");
//Add the time stamp by using the server URI
signature.TimestampServer = new TimestampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new pdf document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPage page = document.Pages.Add();
//Creates a digital signature
PdfSignature signature = new PdfSignature(page, "Signature");
//Add the time stamp by using the server URI
signature.TimestampServer = new TimestampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document
```

```
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Creates a digital signature
PdfSignature signature = new PdfSignature(page, "Signature");
//Add the time stamp by using the server URI
signature.TimeStampServer = new TimeStampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}
```

Adding a timestamp to existing PDF document

You can add timestamp to the existing PDF document using the following code snippet.

C#

```
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Add a new page
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Creates a digital signature
PdfSignature signature = new PdfSignature(page, "Signature");
//Add the time stamp by using the server URI
signature.TimeStampServer = new TimeStampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
//Saves the document
loadedDocument.Save("Output.pdf");
//Closes the document
```

```
loadedDocument.Close(true);
```

VB.NET

```
'Load the PDF document
Dim document As New PdfLoadedDocument("Input.pdf")
'Gets the first page of the document
Dim page As PdfLoadedPage = TryCast(document.Pages(0), PdfLoadedPage)
'Creates a digital signature
Dim signature As New PdfSignature(page, "Signature")
'Adds time stamp by using the server URI
signature.TimestampServer = New TimestampServer(New
Uri("http://syncfusion.digistamp.com"), "user", "123456")
'Saves and closes the document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Gets the first page of the document
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Creates a digital signature
PdfSignature signature = new PdfSignature(page, "Signature");
//Add the time stamp by using the server URI
signature.TimestampServer = new TimestampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Gets the page
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Creates a digital signature.
PdfSignature signature = new PdfSignature(page, "Signature");
//Add the time stamp by using the server URI
```



```
signature.TimeStampServer = new TimeStampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Gets the page
PdfLoadedPage page = loadedDocument.Pages[0] as PdfLoadedPage;
//Creates a digital signature
PdfSignature signature = new PdfSignature(page, "Signature");
//Add the time stamp by using the server URI
signature.TimeStampServer = new TimeStampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456")
//Save the PDF document to stream
MemoryStream memoryStream = new MemoryStream();
loadedDocument.Save(memoryStream);
//Closes the document
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}
```

Retrieve certificate details from an existing signed PDF document

The Essential PDF provides support to get the certificate details from an existing signed PDF document such as,

- Signed date
- Expiry date
- Signed name
- Subject name
- Issuer name
- Certificate distinguished names (country, state, street, email, organization, organization unit, locality, and more).

You can get the above certificate details from an existing signed PDF document by using the following code snippet.

C#

```
//Load the existing signed PDF
PdfLoadedDocument loadedDocument = new
PdfLoadedDocument("SignedDocument.pdf");
//Load the PDF form
PdfLoadedForm loadedForm = loadedDocument.Form as PdfLoadedForm;
//Get the signature field
PdfLoadedSignatureField signatureField = loadedForm.Fields[0] as
PdfLoadedSignatureField;
//Get the certificate
PdfCertificate certificate = signatureField.Signature.Certificate;
//Get the signed date
DateTime date = signatureField.Signature.SignedDate;
//Get the signed name
string name = signatureField.Signature.SignedName;
//Get the certificate names based on their distinguished name.
string subjectName = certificate.SubjectName;
string subjectCountry =
certificate.GetValue(PdfCertificateDistinguishedName.Country,
PdfCertificateField.Subject);
string subjectOrganization =
certificate.GetValue(PdfCertificateDistinguishedName.Organization,
PdfCertificateField.Subject);
//Issuer details
string issuerName = certificate.IssuerName;
string issuerOrganization =
certificate.GetValue(PdfCertificateDistinguishedName.Organization,
PdfCertificateField.Issuer);
string issuerCountry =
certificate.GetValue(PdfCertificateDistinguishedName.Country,
PdfCertificateField.Issuer);
//Get certificate validation date information
DateTime validFrom = certificate.ValidFrom;
DateTime validTo = certificate.ValidTo;
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing signed PDF
Dim loadedDocument As PdfLoadedDocument = New
PdfLoadedDocument("../Signed.pdf")
'Load the PDF form
```

```

Dim loadedForm As PdfLoadedForm = TryCast(loadedDocument.Form,
PdfLoadedForm)
'Get the signature field
Dim signatureField As PdfLoadedSignatureField =
TryCast(loadedForm.Fields(0), PdfLoadedSignatureField)
'Get the certificate
Dim certificate As PdfCertificate = signatureField.Signature.Certificate
'Get the signed date
Dim signedDate As DateTime = signatureField.Signature.SignedDate
'Get the signed name
Dim name As String = signatureField.Signature.SignedName
'Get the certificate names based on their distinguished name
Dim subjectName As String = certificate.SubjectName
Dim subjectCountry As String =
certificate.GetValue(PdfCertificateDistinguishedName.Country,
PdfCertificateField.Subject)
Dim subjectOrganization As String =
certificate.GetValue(PdfCertificateDistinguishedName.Organization,
PdfCertificateField.Subject)
'Issuer details
Dim issuerName As String = certificate.IssuerName
Dim issuerOrganization As String =
certificate.GetValue(PdfCertificateDistinguishedName.Organization,
PdfCertificateField.Issuer)
Dim issuerCountry As String =
certificate.GetValue(PdfCertificateDistinguishedName.Country,
PdfCertificateField.Issuer)
'Get certificate validation date information
Dim validFrom As DateTime = certificate.ValidFrom
Dim validTo As DateTime = certificate.ValidTo
loadedDocument.Close(True)

```

UWP

```

//Essential PDF supports retrieving certificate details from signed PDF
document only in Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Get the signature field
PdfLoadedSignatureField signatureField = loadedDocument.Form.Fields[0] as
PdfLoadedSignatureField;
//Get the certificate
PdfCertificate certificate = signatureField.Signature.Certificate;
//Get the signed date
DateTime date = signatureField.Signature.SignedDate;
//Get the signed name
string name = signatureField.Signature.SignedName;
//Gets the certificate subject's name
string subjectName = certificate.SubjectName;
//Gets the certificate issuer's name
string issuerName = certificate.IssuerName;

```

```
//Get certificate validation date information
DateTime validFrom = certificate.ValidFrom;
DateTime validTo = certificate.ValidTo;
//Close the document
loadedDocument.Close(true);
```

XAMARIN

```
//Essential PDF supports retrieving certificate details from signed PDF
document only in Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms
```

Enable Long Term Validation (LTV) PDF signature

The Essential PDF supports creating long term signature validation when signing the PDF document. The LTV signature allows you to check the validity of a signature long after the document was signed. To achieve long term validation, all the required elements for signature validation must be embedded in the signed PDF.

Note: The resulted PDF document size will be large since all the necessary signature information, Certificate Revocation List (CRL), and Online Certificate Status Protocol (OCSP) are embedded.

The following code example explains how to create LTV PDF.

C#

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
PdfCertificate pdfCert = new PdfCertificate(@"PDF.pfx", "syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(page, pdfCert, "Signature");
//Sets an image for signature field
PdfBitmap image = new PdfBitmap(@"syncfusion_logo.gif");
//Adds time stamp by using the server URI and credentials
signature.TimeStampServer = new TimeStampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
//Enable LTV on Signature
signature.EnableLtv = true;
//Sets signature info
signature.Bounds = new RectangleF(new PointF(0, 0),
image.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(image, 0, 0);
//Saves and closes the document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
```

```

Dim document As New PdfDocument()
'Adds a new page
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
'Creates a certificate instance from PFX file with private key
Dim pdfCert As New PdfCertificate("PDF.pfx", "syncfusion")
'Creates a digital signature
Dim signature As New PdfSignature(page, pdfCert, "Signature")
'Sets an image for signature field
Dim image As New PdfBitmap("syncfusion_logo.gif")
'Adds time stamp by using the server URI and credentials
signature.TimeStampServer = New TimeStampServer(New
Uri("http://syncfusion.digistamp.com"), "user", "123456")
'Enable LTV on Signature
signature.EnableLtv = True
'Sets signature info
signature.Bounds = New RectangleF(New PointF(0, 0), image.PhysicalDimension)
signature.ContactInfo = "johndoe@owned.us"
signature.LocationInfo = "Honolulu, Hawaii"
signature.Reason = "I am author of this document."
'Draws the signature image
graphics.DrawImage(image, 0, 0)
'Saves and closes the document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.PDF.pfx");
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.syncfusion_logo.gif");
PdfBitmap image = new PdfBitmap(imageStream);
//Adds time stamp by using the server URI and credentials
signature.TimeStampServer = new TimeStampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
//Enable LTV on Signature
signature.EnableLtv = true;
//Sets signature info
signature.Bounds = new RectangleF(new PointF(0, 0),
image.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";

```

```
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(image, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
FileStream certificateStream = new FileStream("PDF.pfx", FileMode.Open,
FileAccess.Read);
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
FileStream imageStream = new FileStream("syncfusion_logo.gif",
FileMode.Open, FileAccess.Read);
//Sets an image for signature field
PdfBitmap image = new PdfBitmap(imageStream);
//Adds time stamp by using the server URI and credentials
signature.TimestampServer = new TimestampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
//Enable LTV on Signature
signature.EnableLtv = true;
//Sets signature info
signature.Bounds = new RectangleF(new PointF(0, 0),
image.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(image, 0, 0);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
```

```
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF.pfx");
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets an image for signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.syncfusion_logo.gif");
PdfBitmap image = new PdfBitmap(imageStream);
//Adds time stamp by using the server URI and credentials
signature.TimestampServer = new TimestampServer(new
Uri("http://syncfusion.digistamp.com"), "user", "123456");
//Enable LTV on Signature
signature.EnableLtv = true;
//Sets signature info
signature.Bounds = new RectangleF(new PointF(0, 0),
image.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(image, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
"application/pdf", stream);
}
else
{
```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
"application/pdf", stream);
}
```

Adding a digital signature with customization

The [PdfSignatureSettings](#) allows you to add customized digital signatures to the PDF document.

Adding a digital signature with CADES format

As per the PDF specification 2.0, now Syncfusion PDF library supports digital signature based on CADES (CMS Advanced Electronics Signature). The CADES based digital signature can remain valid for long periods, even if underlying cryptographic algorithms are broken. Using the API [CryptographicStandard](#), you can change the standard between CMS (Cryptographic Message Syntax) and CADES.

The following code example explains how to add a digital signature with cryptographic standard (CADES) to the PDF document.

C#

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
PdfCertificate pdfCert = new PdfCertificate(@"PDF.pfx", "syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets signature settings to customize cryptographic standard specified
PdfSignatureSettings settings = signature.Settings;
settings.CryptographicStandard = CryptographicStandard.CADES;
//Sets an image for signature field
PdfBitmap signatureImage = new PdfBitmap(@"signature.jpg");
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Saves and closes the document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Adds a new page
Dim page As PdfPageBase = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
'Creates a certificate instance from PFX file with private key
Dim pdfCert As New PdfCertificate("PDF.pfx", "syncfusion")
'Creates a digital signature
```



```

Dim signature As New PdfSignature(document, page, pdfCert, "Signature")
'Sets signature settings to customize cryptographic standard specified
Dim settings As PdfSignatureSettings = signature.Settings
settings.CryptographicStandard = CryptographicStandard.CADES
'Sets an image for signature field
Dim signatureImage As New PdfBitmap("signature.jpg")
'Sets signature info
signature.Bounds = New RectangleF(New PointF(0, 0),
signatureImage.PhysicalDimension)
signature.ContactInfo = "johndoe@owned.us"
signature.LocationInfo = "Honolulu, Hawaii"
signature.Reason = "I am author of this document."
'Draws the signature image
graphics.DrawImage(signatureImage, 0, 0)
'Saves and closes the document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.PDF.pfx");
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets signature settings to customize cryptographic standard specified
PdfSignatureSettings settings = signature.Settings;
settings.CryptographicStandard = CryptographicStandard.CADES;
//Sets an image for signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.signature.jpg");
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);

```

```
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
FileStream certificateStream = new FileStream("PDF.pfx", FileMode.Open,
FileAccess.Read);
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets signature settings to customize cryptographic standard specified
PdfSignatureSettings settings = signature.Settings;
settings.CryptographicStandard = CryptographicStandard.CADES;
//Sets an image for signature field
FileStream imageStream = new FileStream("signature.jpg", FileMode.Open,
FileAccess.Read);
//Sets an image for signature field
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
```

```

//Creates a certificate instance from PFX file with private key
Stream certificateStream =
    typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF.pfx");
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
    "syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
    "Signature");
//Sets signature settings to customize cryptographic standard specified
PdfSignatureSettings settings = signature.Settings;
settings.CryptographicStandard = CryptographicStandard.CADES;
//Sets an image for signature field
Stream imageStream =
    typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.signature.jpg");
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
    signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
        "application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
        "application/pdf", stream);
}

```

Customize digestion algorithm

In addition, you can now set the different message digest algorithm to sign PDF document using the [DigestAlgorithm](#) enum available in the class [PdfSignatureSettings](#).

The following message digest algorithms are now supported:

- SHA1
- SHA256
- SHA384

- SHA512
- RIPEMD160

The following code example explains how to add a digital signature with various digest algorithms to the PDF document.

C#

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
PdfCertificate pdfCert = new PdfCertificate(@"PDF.pfx", "syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets signature settings to customize digest algorithm specified
PdfSignatureSettings settings = signature.Settings;
settings.DigestAlgorithm = DigestAlgorithm.SHA256;
//Sets an image for signature field
PdfBitmap signatureImage = new PdfBitmap(@"signature.jpg");
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Saves and closes the document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim document As New PdfDocument()
'Adds a new page
Dim page As PdfPageBase = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
'Creates a certificate instance from PFX file with private key
Dim pdfCert As New PdfCertificate("PDF.pfx", "syncfusion")
'Creates a digital signature
Dim signature As New PdfSignature(document, page, pdfCert, "Signature")
'Sets signature settings to customize digest algorithm specified
Dim settings As PdfSignatureSettings = signature.Settings
settings.DigestAlgorithm = DigestAlgorithm.SHA256
'Sets an image for signature field
Dim signatureImage As New PdfBitmap("signature.jpg")
'Sets signature info
signature.Bounds = New RectangleF(New PointF(0, 0),
signatureImage.PhysicalDimension)
signature.ContactInfo = "johndoe@owned.us"
signature.LocationInfo = "Honolulu, Hawaii"
```

```
signature.Reason = "I am author of this document."
'Draws the signature image
graphics.DrawImage(signatureImage, 0, 0)
'Saves and closes the document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.PDF.pfx");
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
"syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
"Signature");
//Sets signature settings to customize digest algorithm specified
PdfSignatureSettings settings = signature.Settings;
settings.DigestAlgorithm = DigestAlgorithm.SHA256;
//Sets an image for signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.signature.jpg");
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
```

```

FileStream certificateStream = new FileStream("PDF.pfx", FileMode.Open,
    FileAccess.Read);
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
    "syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
    "Signature");
//Sets signature settings to customize digest algorithm specified
PdfSignatureSettings settings = signature.Settings;
settings.DigestAlgorithm = DigestAlgorithm.SHA256;
//Sets an image for signature field
FileStream imageStream = new FileStream("signature.jpg", FileMode.Open,
    FileAccess.Read);
//Sets an image for signature field
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
    signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the documents
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Adds a new page
PdfPageBase page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
    typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF.pfx");
PdfCertificate pdfCert = new PdfCertificate(certificateStream,
    "syncfusion");
//Creates a digital signature
PdfSignature signature = new PdfSignature(document, page, pdfCert,
    "Signature");
//Sets signature settings to customize digest algorithm specified
PdfSignatureSettings settings = signature.Settings;
settings.DigestAlgorithm = DigestAlgorithm.SHA256;

```

```

//Sets an image for signature field
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.signature.jpg");
PdfBitmap signatureImage = new PdfBitmap(imageStream);
//Sets signature information
signature.Bounds = new RectangleF(new PointF(0, 0),
signatureImage.PhysicalDimension);
signature.ContactInfo = "johndoe@owned.us";
signature.LocationInfo = "Honolulu, Hawaii";
signature.Reason = "I am author of this document.";
//Draws the signature image
graphics.DrawImage(signatureImage, 0, 0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Digital signature validation

Added the support to validate the digital signatures in an existing PDF document. Digital signature validation covers the following steps to ensure the validity of the signatures:

- Validate the document modification.
- Validate the certificate chain.
- Ensure the signature with timestamp time.
- Check the revocation status of the certificate with OCSP and CRL.
- Ensure the multiple digital signatures.

You can use the [ValidateSignature](#) method available in the [PdfLoadedSignatureField](#) class to validate the digital signature.

You can get the overall status from the [IsSignatureValid](#) property available in the [PdfSignatureValidationResult](#) class.

The following code example explains how to validate the digitally signed PDF document signature.

C#

```

//Load an existing signed PDF document

```

```

PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Get signature field
PdfLoadedSignatureField signatureField = loadedDocument.Form.Fields[0] as
PdfLoadedSignatureField;
//X509Certificate2Collection to check the signer's identity using root
certificates
X509CertificateCollection collection = new X509CertificateCollection();
//Create new X509Certificate2 with the root certificate
X509Certificate2 certificate = new X509Certificate2("PDF.pfx",
"syncfusion");
//Add the certificate to the collection
collection.Add(certificate);
//Validate signature and get the validation result
PdfSignatureValidationResult result =
signatureField.ValidateSignature(collection);
//Checks whether the signature is valid or not
SignatureStatus status = result.SignatureStatus;
//Checks whether the document is modified or not
bool isModified = result.IsDocumentModified;
//Signature details
string issuerName = signatureField.Signature.Certificate.IssuerName;
DateTime validFrom = signatureField.Signature.Certificate.ValidFrom;
DateTime validTo = signatureField.Signature.Certificate.ValidTo;
string signatureAlgorithm = result.SignatureAlgorithm;
DigestAlgorithm digestAlgorithm = result.DigestAlgorithm;
//Revocation validation details
RevocationResult revocationDetails = result.RevocationResult;
RevocationStatus revocationStatus = revocationDetails.OcspRevocationStatus;
bool isRevokedCRL = revocationDetails.IsRevokedCRL;
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load an existing signed PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get signature field
Dim signatureField As PdfLoadedSignatureField =
loadedDocument.Form.Fields[0] As PdfLoadedSignatureField
'X509Certificate2Collection to check the signer's identity using root
certificates
Dim collection As X509CertificateCollection = New
X509CertificateCollection()
'Create new X509Certificate2 with the root certificate
Dim certificate As X509Certificate2 = New X509Certificate2("PDF.pfx",
"syncfusion")
'Add the certificate to the collection
collection.Add(certificate)
'Validate signature and get the validation result
Dim result As PdfSignatureValidationResult =
signatureField.ValidateSignature(collection)
'Checks whether the signature is valid or not
Dim status As SignatureStatus = result.SignatureStatus
'Checks whether the document is modified or not
Dim isModified As Boolean = result.IsDocumentModified
'Signature details

```



```

Dim issuerName As String = signatureField.Signature.Certificate.IssuerName
Dim validFrom As DateTime = signatureField.Signature.Certificate.ValidFrom
Dim validTo As DateTime = signatureField.Signature.Certificate.ValidTo
Dim signatureAlgorithm As String = result.SignatureAlgorithm
Dim digestAlgorithm As DigestAlgorithm = result.DigestAlgorithm
'Revocation validation details
Dim revocationDetails As RevocationResult = result.RevocationResult
Dim revocationStatus As RevocationStatus =
revocationDetails.OcspRevocationStatus
Dim isRevokedCRL As Boolean = revocationDetails.IsRevokedCRL
'Close the document
loadedDocument.Close(true)

```

UWP

```

//Get the stream from the document
Stream documentStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.pdf");
//Load an existing signed PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(documentStream);
//Get signature field
PdfLoadedSignatureField signatureField = loadedDocument.Form.Fields[0] as
PdfLoadedSignatureField;
//X509Certificate2Collection to check the signer's identity using root
certificates
X509CertificateCollection collection = new X509CertificateCollection();
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.PDF.pfx");
byte[] data = new byte[certificateStream.Length];
certificateStream.Read(data, 0, data.Length);
//Create new X509Certificate2 with the root certificate
X509Certificate2 certificate = new X509Certificate2(data, "syncfusion");
//Add the certificate to the collection
collection.Add(certificate);
//Validate signature and get the validation result
PdfSignatureValidationResult result =
signatureField.ValidateSignature(collection);
//Checks whether the signature is valid or not
SignatureStatus status = result.SignatureStatus;
//Checks whether the document is modified or not
bool isModified = result.IsDocumentModified;
//Signature details
string issuerName = signatureField.Signature.Certificate.IssuerName;
DateTime validFrom = signatureField.Signature.Certificate.ValidFrom;
DateTime validTo = signatureField.Signature.Certificate.ValidTo;
string signatureAlgorithm = result.SignatureAlgorithm;
DigestAlgorithm digestAlgorithm = result.DigestAlgorithm;
//Revocation validation details
RevocationResult revocationDetails = result.RevocationResult;
RevocationStatus revocationStatus = revocationDetails.OcspRevocationStatus;
bool isRevokedCRL = revocationDetails.IsRevokedCRL;
//Close the document
loadedDocument.Close(true);

```

ASP.NET CORE

```
//Get the stream from the document
FileStream documentStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
//Load an existing signed PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(documentStream);
//Get signature field
PdfLoadedSignatureField signatureField = loadedDocument.Form.Fields[0] as
PdfLoadedSignatureField;
//X509Certificate2Collection to check the signer's identity using root
certificates
X509CertificateCollection collection = new X509CertificateCollection();
//Creates a certificate instance from PFX file with private key
FileStream certificateStream = new FileStream("PDF.pfx", FileMode.Open,
FileAccess.Read);
byte[] data = new byte[certificateStream.Length];
certificateStream.Read(data, 0, data.Length);
//Create new X509Certificate2 with the root certificate
X509Certificate2 certificate = new X509Certificate2(data, "syncfusion");
//Add the certificate to the collection
collection.Add(certificate);
//Validate signature and get the validation result
PdfSignatureValidationResult result =
signatureField.ValidateSignature(collection);
//Checks whether the signature is valid or not
SignatureStatus status = result.SignatureStatus;
//Checks whether the document is modified or not
bool isModified = result.IsDocumentModified;
//Signature details
string issuerName = signatureField.Signature.Certificate.IssuerName;
DateTime validFrom = signatureField.Signature.Certificate.ValidFrom;
DateTime validTo = signatureField.Signature.Certificate.ValidTo;
string signatureAlgorithm = result.SignatureAlgorithm;
DigestAlgorithm digestAlgorithm = result.DigestAlgorithm;
//Revocation validation details
RevocationResult revocationDetails = result.RevocationResult;
RevocationStatus revocationStatus = revocationDetails.OcspRevocationStatus;
bool isRevokedCRL = revocationDetails.IsRevokedCRL;
//Close the document
loadedDocument.Close(true);
```

XAMARIN

```
//Get the stream from the document
Stream documentStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
//Load an existing signed PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(documentStream);
//Get signature field
PdfLoadedSignatureField signatureField = loadedDocument.Form.Fields[0] as
PdfLoadedSignatureField;
//X509Certificate2Collection to check the signer's identity using root
certificates
```

```

X509CertificateCollection collection = new X509CertificateCollection();
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF.pfx");
byte[] data = new byte[certificateStream.Length];
certificateStream.Read(data, 0, data.Length);
//Create new X509Certificate2 with the root certificate
X509Certificate2 certificate = new X509Certificate2(data, "syncfusion");
//Add the certificate to the collection
collection.Add(certificate);
//Validate signature and get the validation result
PdfSignatureValidationResult result =
signatureField.ValidateSignature(collection);
//Checks whether the signature is valid or not
SignatureStatus status = result.SignatureStatus;
//Checks whether the document is modified or not
bool isModified = result.IsDocumentModified;
//Signature details
string issuerName = signatureField.Signature.Certificate.IssuerName;
DateTime validFrom = signatureField.Signature.Certificate.ValidFrom;
DateTime validTo = signatureField.Signature.Certificate.ValidTo;
string signatureAlgorithm = result.SignatureAlgorithm;
DigestAlgorithm digestAlgorithm = result.DigestAlgorithm;
//Revocation validation details
RevocationResult revocationDetails = result.RevocationResult;
RevocationStatus revocationStatus = revocationDetails.OcspRevocationStatus;
bool isRevokedCRL = revocationDetails.IsRevokedCRL;
//Close the document
loadedDocument.Close(true);

```

Validate all signatures in PDF document

Added the support to validate all the digital signatures in an existing PDF document.

You can use the `ValidateSignatures` method available in the [PdfLoadedFormFieldCollection](#) class to validate all the digital signatures. You can get the list of [PdfSignatureValidationResult](#) from the `ValidateSignatures` method.

The following code example explains how to validate all the signatures in digitally signed PDF document.

C#

```

//Load an existing signed PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//X509Certificate2Collection to check the signer's identity using root
certificates
X509CertificateCollection collection = new X509CertificateCollection();
//Create new X509Certificate2 with the root certificate
X509Certificate2 certificate = new X509Certificate2("PDF.pfx",
"syncfusion");
//Add the certificate to the collection
collection.Add(certificate);
//Validate all signatures in loaded PDF document and get the list of
validation result
List<PdfSignatureValidationResult> results;

```

```

bool isValid = loadedDocument.Form.Fields.ValidateSignatures(collection, out
results);
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load an existing signed PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'X509Certificate2Collection to check the signer's identity using root
certificates
Dim collection As X509CertificateCollection = New
X509CertificateCollection()
'Create new X509Certificate2 with the root certificate
Dim certificate As X509Certificate2 = New X509Certificate2("PDF.pfx",
"syncfusion")
'Add the certificate to the collection
collection.Add(certificate)
'Validate all signatures in loaded PDF document and get the list of
validation result
Dim results As List<PdfSignatureValidationResult>
Dim isValid As Boolean =
loadedDocument.Form.Fields.ValidateSignatures(collection, out results)
'Close the document
loadedDocument.Close(true)

```

UWP

```

//Get the stream from the document
Stream documentStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.pdf");
//Load an existing signed PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(documentStream);
//X509Certificate2Collection to check the signer's identity using root
certificates
X509CertificateCollection collection = new X509CertificateCollection();
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.PDF.pfx");
byte[] data = new byte[certificateStream.Length];
certificateStream.Read(data, 0, data.Length);
//Create new X509Certificate2 with the root certificate
X509Certificate2 certificate = new X509Certificate2(data, "syncfusion");
//Add the certificate to the collection
collection.Add(certificate);
//Validate all signatures in loaded PDF document and get the list of
validation result
List<PdfSignatureValidationResult> results;
bool isValid = loadedDocument.Form.Fields.ValidateSignatures(collection, out
results);
//Close the document
loadedDocument.Close(true);

```

ASP.NET CORE

```
//Get the stream from the document
FileStream documentStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
//Load an existing signed PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(documentStream);
//X509Certificate2Collection to check the signer's identity using root
certificates
X509CertificateCollection collection = new X509CertificateCollection();
//Creates a certificate instance from PFX file with private key
FileStream certificateStream = new FileStream("PDF.pfx", FileMode.Open,
FileAccess.Read);
byte[] data = new byte[certificateStream.Length];
certificateStream.Read(data, 0, data.Length);
//Create new X509Certificate2 with the root certificate
X509Certificate2 certificate = new X509Certificate2(data, "syncfusion");
//Add the certificate to the collection
collection.Add(certificate);
//Validate all signatures in loaded PDF document and get the list of
validation result
List<PdfSignatureValidationResult> results;
bool isValid = loadedDocument.Form.Fields.ValidateSignatures(collection, out
results);
//Close the document
loadedDocument.Close(true);
```

XAMARIN

```
//Get the stream from the document
Stream documentStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
//Load an existing signed PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(documentStream);
//X509Certificate2Collection to check the signer's identity using root
certificates
X509CertificateCollection collection = new X509CertificateCollection();
//Creates a certificate instance from PFX file with private key
Stream certificateStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.PDF.pfx");
byte[] data = new byte[certificateStream.Length];
certificateStream.Read(data, 0, data.Length);
//Create new X509Certificate2 with the root certificate
X509Certificate2 certificate = new X509Certificate2(data, "syncfusion");
//Add the certificate to the collection
collection.Add(certificate);
//Validate all signatures in loaded PDF document and get the list of
validation result
List<PdfSignatureValidationResult> results;
bool isValid = loadedDocument.Form.Fields.ValidateSignatures(collection, out
results);
//Close the document
loadedDocument.Close(true);
```

Working with Barcode

Essential PDF provides support to add barcodes to the PDF document. The following barcode types are supported.

- 10 one-dimensional barcodes including Code 39 and Code 32 barcodes.
- 2 two-dimensional barcodes such as QR and DataMatrix barcode

Adding a one dimensional barcode to the PDF document

The below code snippet shows how to add Code39 barcode using the [PdfCode39Barcode](#) class to a PDF document.

C#

```
//Creating new PDF Document
PdfDocument doc = new PdfDocument();
//Adding new page to PDF document
PdfPage page = doc.Pages.Add();
//Drawing Code39 barcode
PdfCode39Barcode barcode = new PdfCode39Barcode();
//Setting height of the barcode
barcode.BarHeight = 45;
barcode.Text = "CODE39$";
//Printing barcode on to the Pdf.
barcode.Draw(page, new PointF(25, 70));
//Saving the Document
doc.Save("CODE39.pdf");
```

VB.NET

```
'Creating new PDF Document
Dim doc As New PdfDocument()
'Adding new page to PDF document
Dim page As PdfPage = doc.Pages.Add()
'Drawing Code39 barcode
Dim barcode As New PdfCode39Barcode()
'Setting height of the barcode
barcode.BarHeight = 45
barcode.Text = "CODE39$"
'Printing barcode on to the Pdf.
barcode.Draw(page, New PointF(25, 70))
'Saving the Document
doc.Save("CODE39.pdf")
```

UWP

```
//Creating new PDF Document
PdfDocument doc = new PdfDocument();
//Adding new page to PDF document
PdfPage page = doc.Pages.Add();
//Drawing Code39 barcode
PdfCode39Barcode barcode = new PdfCode39Barcode();
//Setting height of the barcode
barcode.BarHeight = 45;
barcode.Text = "CODE39$";
```

```
//Printing barcode on to the Pdf.
barcode.Draw(page, new PointF(25, 70));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Creating new PDF Document
PdfDocument doc = new PdfDocument();
//Adding new page to PDF document
PdfPage page = doc.Pages.Add();
//Drawing Code39 barcode
PdfCode39Barcode barcode = new PdfCode39Barcode();
//Setting height of the barcode
barcode.BarHeight = 45;
barcode.Text = "CODE39$";
//Printing barcode on to the Pdf.
barcode.Draw(page, new PointF(25, 70));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
doc.Save(stream);
stream.Position = 0;
//Close the documents.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = " CODE39.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creating new PDF Document
PdfDocument doc = new PdfDocument();
//Adding new page to PDF document
PdfPage page = doc.Pages.Add();
//Drawing Code39 barcode
PdfCode39Barcode barcode = new PdfCode39Barcode();
//Setting height of the barcode
barcode.BarHeight = 45;
barcode.Text = "CODE39$";
//Printing barcode on to the Pdf.
barcode.Draw(page, new PointF(25, 70));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Closes the document
doc.Close(true);
```

```
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Adding a two dimensional barcode to a PDF document

The below code snippet shows how to add a QR code using [PdfQRBarcode](#) class to the PDF document.

C#

```
//Drawing QR Barcode
PdfQRBarcode barcode = new PdfQRBarcode();
//Set Error Correction Level
barcode.ErrorCorrectionLevel = PdfErrorCorrectionLevel.High;
//Set XDimension
barcode.XDimension = 3;
barcode.Text = "http://www.syncfusion.com";
//Creating new PDF Document
PdfDocument doc = new PdfDocument();
//Adding new page to PDF document
PdfPage page = doc.Pages.Add();
//Printing barcode on to the Pdf.
barcode.Draw(page, new PointF(25, 70));
//Saving the Document
doc.Save("QRBarcode.pdf");
```

VB.NET

```
'Drawing QR Barcode
Dim barcode As New PdfQRBarcode()
'Set Error Correction Level
barcode.ErrorCorrectionLevel = PdfErrorCorrectionLevel.High
'Set XDimension
barcode.XDimension = 3
barcode.Text = "http://www.syncfusion.com"
'Creating new PDF Document
Dim doc As New PdfDocument()
'Adding new page to PDF document
Dim page As PdfPage = doc.Pages.Add()
'Printing barcode on to the Pdf.
barcode.Draw(page, New PointF(25, 70))
'Saving the Document
doc.Save("QRBarcode.pdf")
```


UWP

```
//Drawing QR Barcode
PdfQRBarcode barcode = new PdfQRBarcode();
//Set Error Correction Level
barcode.ErrorCorrectionLevel = PdfErrorCorrectionLevel.High;
//Set XDimension
barcode.XDimension = 3;
barcode.Text = "http://www.syncfusion.com";
//Creating new PDF Document
PdfDocument doc = new PdfDocument();
//Adding new page to PDF document
PdfPage page = doc.Pages.Add();
//Printing barcode on to the Pdf.
barcode.Draw(page, new PointF(25, 70));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Drawing QR Barcode
PdfQRBarcode barcode = new PdfQRBarcode();
//Set Error Correction Level
barcode.ErrorCorrectionLevel = PdfErrorCorrectionLevel.High;
//Set XDimension
barcode.XDimension = 3;
barcode.Text = "http://www.syncfusion.com";
barcode.Text = "http://www.syncfusion.com";
//Creating new PDF Document
PdfDocument doc = new PdfDocument();
//Adding new page to PDF document
PdfPage page = doc.Pages.Add();
//Printing barcode on to the Pdf.
barcode.Draw(page, new PointF(25, 70));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
doc.Save(stream);
stream.Position = 0;
//Close the documents.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "QRBarcode.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Drawing QR Barcode
PdfQRBarcode barcode = new PdfQRBarcode();
//Set Error Correction Level
barcode.ErrorCorrectionLevel = PdfErrorCorrectionLevel.High;
//Set XDimension
barcode.XDimension = 3;
barcode.Text = "http://www.syncfusion.com";
//Creating new PDF Document
PdfDocument doc = new PdfDocument();
//Adding new page to PDF document
PdfPage page = doc.Pages.Add();
//Printing barcode on to the Pdf.
barcode.Draw(page, new PointF(25, 70));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Closes the document
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Set location and size to the barcode

The following code snippets show how to set [Size](#) and [Location](#) for Codabar barcode using [PdfCodabarBarcode](#) class to a PDF document.

C#

```
//Creating new PDF Document
PdfDocument doc = new PdfDocument();
//Adding new page to PDF document
PdfPage page = doc.Pages.Add();
//Create new instance for Codabar barcode
PdfCodabarBarcode barcode = new PdfCodabarBarcode();
//Setting location of the barcode
barcode.Location = new PointF(100, 100);
//Setting size of the barcode
barcode.Size = new SizeF(200, 100);
barcode.Text = "123456789$";
//Printing barcode on to the Pdf.
barcode.Draw(page);
//Save and close the Document
doc.Save("CODABAR.pdf");
doc.Close(true);
```

VB.NET

```

'Creating new PDF Document
Dim doc As New PdfDocument()
'Adding new page to PDF document
Dim page As PdfPage = doc.Pages.Add()
'Create new instance for Codabar barcode
Dim barcode As PdfCodabarBarcode = New PdfCodabarBarcode()
'Setting location of the barcode
barcode.Location = New PointF(100, 100)
'Setting size of the barcode
barcode.Size = New SizeF(200, 100)
barcode.Text = "123456789$"
'Printing barcode on to the Pdf.
barcode.Draw(page)
'Save and close the Document
doc.Save("CODABAR.pdf")
doc.Close(True)

```

UWP

```

//Creating new PDF Document
PdfDocument doc = new PdfDocument();
//Adding new page to PDF document
PdfPage page = doc.Pages.Add();
//Create new instance for Codabar barcode
PdfCodabarBarcode barcode = new PdfCodabarBarcode();
//Setting location of the barcode
barcode.Location = new PointF(100, 100);
//Setting size of the barcode
barcode.Size = new SizeF(200, 100);
barcode.Text = "123456789$";
//Printing barcode on to the Pdf.
barcode.Draw(page);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Creating new PDF Document
PdfDocument doc = new PdfDocument();
//Adding new page to PDF document
PdfPage page = doc.Pages.Add();
//Create new instance for Codabar barcode
PdfCodabarBarcode barcode = new PdfCodabarBarcode();
//Setting location of the barcode
barcode.Location = new PointF(100, 100);
//Setting size of the barcode

```

```

barcode.Size = new SizeF(200, 100);
barcode.Text = "123456789$";
//Printing barcode on to the Pdf.
barcode.Draw(page);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
doc.Save(stream);
stream.Position = 0;
//Close the documents.
doc.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = " CODABAR.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creating new PDF Document
PdfDocument doc = new PdfDocument();
//Adding new page to PDF document
PdfPage page = doc.Pages.Add();
//Create new instance for Codabar barcode
PdfCodabarBarcode barcode = new PdfCodabarBarcode();
//Setting location of the barcode
barcode.Location = new PointF(100, 100);
//Setting size of the barcode
barcode.Size = new SizeF(200, 100);
barcode.Text = "123456789$";
//Printing barcode on to the Pdf.
barcode.Draw(page);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Closes the document
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Export Barcode as Image

Essential PDF supports converting one dimensional barcodes such as Code 39, Code 39 Extended, Code 11, Codabar, Code 32, Code 93, Code 93 Extended, Code 128A, Code 128B, UPC bar code, and Code 128C barcodes to image using the [ToImage](#) method of [PdfUnidimensionalBarcode](#) instance.

Note: To export barcode as image in .NET Core, the following assembly should be referenced in your application [Syncfusion.Pdf.Imaging.Portable](#) .

The following code snippet explains this.

C#

```
//Initialize a new PdfCode39Barcode instance
PdfCode39Barcode barcode = new PdfCode39Barcode();
//Set the height and text for barcode
barcode.BarHeight = 45;
barcode.Text = "CODE39$";
//Convert the barcode to image
Image barcodeImage = barcode.ToImage(new SizeF(300, 200));
//Save the image
barcodeImage.Save("Image.png", ImageFormat.Png);
```

VB.NET

```
'Initialize a new PdfCode39Barcode instance
Dim barcode As PdfCode39Barcode = New PdfCode39Barcode
'Set the height and text for barcode
barcode.BarHeight = 45
barcode.Text = "CODE39$"
'Convert the barcode to image
Dim barcodeImage As Image = barcode.ToImage(New SizeF(300, 200))
'Save the image
barcodeImage.Save("Image.png", ImageFormat.Png)
```

UWP

```
//PDF supports conversion of Barcode to Image only in Windows Forms, WPF,
ASP.NET and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//Initialize a new PdfCode39Barcode instance
PdfCode39Barcode barcode = new PdfCode39Barcode();
//Set the height and text for barcode
barcode.BarHeight = 45;
barcode.Text = "CODE39$";
//Convert the barcode to image
Image barcodeImage = barcode.ToImage(new SizeF(300, 200));
using (MemoryStream stream = new MemoryStream())
{
    //Save image to stream.
    barcodeImage.Save(stream, ImageFormat.Png);
}
```

XAMARIN

//PDF supports conversion of Barcode to Image only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.

Essential PDF supports converting two-dimensional barcodes such as QR Code and Data Matrix barcode to image. The following code snippet explains how to convert a QR code to image using the [ToImage](#) method of [PdfQRBarcode](#) instance.

C#

```
//Initialize a new PdfQRBarcode instance
PdfQRBarcode barcode = new PdfQRBarcode();
//Set the XDimension and text for barcode
barcode.XDimension = 3;
barcode.Text = "http://www.google.com";
//Convert the barcode to image
Image barcodeImage = barcode.ToImage(new SizeF(300, 300));
//Save the image
barcodeImage.Save("Image.png", ImageFormat.Png);
```

VB.NET

```
'Initialize a new PdfQRBarcode instance
Dim barcode As PdfQRBarcode = New PdfQRBarcode
'Set the XDimension and text for barcode
barcode.XDimension = 3
barcode.Text = "http://www.google.com"
'Convert the barcode to image
Dim barcodeImage As Image = barcode.ToImage(New SizeF(300, 300))
'Save the image
barcodeImage.Save("Image.jpg", ImageFormat.Png)
```

UWP

//PDF supports conversion of Barcode to Image only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.

ASP.NET CORE

```
//Initialize a new PdfQRBarcode instance
PdfQRBarcode barcode = new PdfQRBarcode();
//Set the XDimension and text for barcode
barcode.XDimension = 3;
barcode.Text = "http://www.google.com";
//Convert the barcode to image
Image barcodeImage = barcode.ToImage(new SizeF(300, 300));
using (MemoryStream stream = new MemoryStream())
{
    //Save image to stream.
    barcodeImage.Save(stream, ImageFormat.Png);
}
```

XAMARIN

//PDF supports conversion of Barcode to Image only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.

Customizing the barcode appearance

The height of the barcode can be changed using the [BarHeight](#) property. The equivalent property to change the block size for two dimensional barcode is [XDimension](#). You can also customize the barcode color by changing the DarkBarColor and LightBarColor properties.

Note: This color customization is possible only for one dimensional barcodes and it is not supported for two dimensional barcodes.

Supported barcode types

The following table contains the supported types and associated valid characters.

Symbol	Supported characters	Length
QR Code	[0-9]; [A-Z (upper-case only)]; [space \$ % * + - . / , :]; [Shift JIS characters]	variable
DataMatrix	All ASCII characters	
Code 39	[0-9]; [A-Z]; [- . \$ / + % SPACE]	variable
Code 39 Extended	[0-9]; [A-Z]; [a-z]	variable
Code 11	[0-9]; [-]	variable
Codabar	[0-9]; [- \$: / . +]	variable
Code 32	[0-9]	8
Code 93	[0-9]; [A-Z]; [- . \$ / + % SPACE]	variable
Code 93 Extended	All 128 ASCII characters	variable
Code 128A	[0-9]; [A-Z]; [NUL (0x00) SOH (0x01) STX (0x02) ETX (0x03) EOT(0x04) ENQ (0x05) ACK (0x06) BEL (0x07) BS (0x08) HT (0x09) LF (0x0A) VT(0x0B) FF (0x0C) CR (0x0D) SO (0x0E) SI (0x0F) DLE (0x10) DC1 (0x11) DC2(0x12) DC3 (0x13) DC4 (0x14) NAK (0x15) SYN (0x16) ETB (0x17) CAN(0x18) EM (0x19) SUB (0x1A) ESC (0x1B) FS (0x1C) GS (0x1D) RS (0x1E) US(0x1F) SPACE (0x20)]; [" ! # \$ % & ' () * + , - . / ; < = > ? @ [/] ^ _]	variable

Symbol	Supported characters	Length
Code 128B	[0-9]; [A-Z]; [a-z]; [SPACE (0x20) ! " # \$ % & ' () * + , - . / : ; < = > ? @ [/] ^ _ ` { } ~ DEL (â€¢)]	variable
Code 128C	ASCII 00-99(encodes each two digit with one code)	variable

Working with Actions

Essential PDF supports different actions that can be triggered by different events and user interactions.

Adding an action to the PDF

You can add the action to the PDF document using the below code snippet.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create and add new launch action to the document
PdfLaunchAction action = new PdfLaunchAction("../Data/logo.png");
document.Actions.AfterOpen = action;
//Save the document
document.Save("LaunchAction.pdf");
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As New PdfDocument()
'Create and add new launch action to the document
Dim action As New PdfLaunchAction("../Data/logo.png")
document.Actions.AfterOpen = action
'Save the document
document.Save("LaunchAction.pdf")
document.Close(True)
```

UWP

```
//PDF supports adding Launch action to the PDF document only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//PDF supports adding Launch action to the PDF document only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```

XAMARIN

```
//PDF supports adding Launch action to the PDF document only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.
```


Supported action types

Essential PDF supports the following types of actions.

- [PdfSoundAction](#) that plays the music file
- [PdfJavaScriptAction](#) that executes PDF JavaScript code
- [PdfUriAction](#) that launches the URI
- [PdfGoToAction](#) that goes to the specified page of the document
- [PdfLaunchAction](#) that launches the application or opens the document
- [PdfNamedAction](#) that goes to the named destination: next, previous, first or last page
- [PdfSubmitAction](#) that submits the data that is entered into the PDF form
- [PdfResetAction](#) that resets the fields of the PDF form

Sound action

[PdfSoundAction](#) plays a specified music file in the PDF document. Volume and repeat can be specified for the sound action.

C#

```
//Create a new document
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create a sound action
PdfSoundAction soundAction = new PdfSoundAction("../Data/Startup.wav");
soundAction.Sound.Bits = 16;
soundAction.Sound.Channels = PdfSoundChannels.Stereo;
soundAction.Sound.Encoding = PdfSoundEncoding.Signed;
soundAction.Volume = 0.9f;
//Set the sound action
document.Actions.AfterOpen = soundAction;
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new document
Dim document As New PdfDocument()
'Add a page.
Dim page As PdfPage = document.Pages.Add()
'Create a sound action
Dim soundAction As New PdfSoundAction("../Data/Startup.wav")
soundAction.Sound.Bits = 16
soundAction.Sound.Channels = PdfSoundChannels.Stereo
soundAction.Sound.Encoding = PdfSoundEncoding.Signed
soundAction.Volume = 0.9F
'Set the sound action
document.Actions.AfterOpen = soundAction
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)
```

UWP

//PDF supports sound action only in Windows Forms, WPF, ASP.NET, ASP.NET MVC, ASP.NET Core and Xamarin platforms.

ASP.NET CORE

```
//Create a new document
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create a sound action
FileStream fileStream = new FileStream("Startup.wav", FileMode.Open,
    FileAccess.Read);
PdfSoundAction soundAction = new PdfSoundAction(fileStream);
soundAction.Sound.Bits = 16;
soundAction.Sound.Channels = PdfSoundChannels.Stereo;
soundAction.Sound.Encoding = PdfSoundEncoding.Signed;
soundAction.Volume = 0.9f;
//Set the sound action
document.Actions.AfterOpen = soundAction;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new document
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create a sound action
Stream stream =
    typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Startup.wav");
PdfSoundAction soundAction = new PdfSoundAction(stream);
soundAction.Sound.Bits = 16;
soundAction.Sound.Channels = PdfSoundChannels.Stereo;
soundAction.Sound.Encoding = PdfSoundEncoding.Signed;
soundAction.Volume = 0.9f;
//Set the sound action
document.Actions.AfterOpen = soundAction;
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream into pdf file
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android
//and iOS platforms. Please refer PDF/Xamarin section for respective code
//samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}
```

JavaScript action

A [PdfJavaScriptAction](#) allows execution of **JavaScript** code embedded in the **PDF** document.

C#

```
//Create a new document
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create JavaScript action
PdfJavaScriptAction scriptAction = new
PdfJavaScriptAction("app.alert(\"Hello World!!!\")");
//Add the JavaScript action
document.Actions.AfterOpen = scriptAction;
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new document
Dim document As New PdfDocument()
'Add a page.
Dim page As PdfPage = document.Pages.Add()
'Create JavaScript action
Dim scriptAction As New PdfJavaScriptAction("app.alert(\"\"Hello World!!!\"")")
'Add the JavaScript action
document.Actions.AfterOpen = scriptAction
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a new document
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create JavaScript action
```

```

PdfJavaScriptAction scriptAction = new
PdfJavaScriptAction("app.alert(\"Hello World!!!\")");
//Add the JavaScript action
document.Actions.AfterOpen = scriptAction;
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(memoryStream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new document
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create JavaScript action
PdfJavaScriptAction scriptAction = new
PdfJavaScriptAction("app.alert(\"Hello World!!!\")");
//Add the JavaScript action
document.Actions.AfterOpen = scriptAction;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new document
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create JavaScript action
PdfJavaScriptAction scriptAction = new
PdfJavaScriptAction("app.alert(\"Hello World!!!\")");
//Add the JavaScript action
document.Actions.AfterOpen = scriptAction;
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream into pdf file

```

```
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}
```

Note: You can refer more PDF JavaScript code in **PdfJavaScriptAction** from the below developer guide.

<http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/jsdeveloperpage.pdf>

URI action

PdfUriAction allows you to create a hyperlink that can open web page in a web browser.

C#

```
//Create a new document with PDF/A standard.
PdfDocument document = new PdfDocument();
//Create a uri action
PdfUriAction uriAction = new PdfUriAction("http://www.google.com");
//Add the action to the document
document.Actions.AfterOpen = uriAction;
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new document with PDF/A standard.
Dim document As New PdfDocument()
'Create a uri action
Dim uriAction As New PdfUriAction("http://www.google.com")
'Add the action to the document
document.Actions.AfterOpen = uriAction
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a new document with PDF/A standard.
PdfDocument document = new PdfDocument();
//Create a uri action
PdfUriAction uriAction = new PdfUriAction("http://www.google.com");
//Add the action to the document
document.Actions.AfterOpen = uriAction;
MemoryStream memoryStream = new MemoryStream();
//Save the document.
```

```
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(memoryStream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new document with PDF/A standard.
PdfDocument document = new PdfDocument();
//Create a uri action
PdfUriAction uriAction = new PdfUriAction("http://www.google.com");
//Add the action to the document
document.Actions.AfterOpen = uriAction;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new document with PDF/A standard.
PdfDocument document = new PdfDocument();
//Create a uri action
PdfUriAction uriAction = new PdfUriAction("http://www.google.com");
//Add the action to the document
document.Actions.AfterOpen = uriAction;
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}
```

```
}

```

GoTo action

[PdfGoToAction](#) displays the specified page in the current document. The location can be specified for the destination page.

C#

```
//Create a new document
PdfDocument document = new PdfDocument();
//Add first page
PdfPage page = document.Pages.Add();
//Add second page
PdfPage secondPage = document.Pages.Add();
//Set the goto action
PdfGoToAction gotoAction = new PdfGoToAction(secondPage);
//Set destination location
gotoAction.Destination = new PdfDestination(secondPage, new PointF(0, 100));
document.Actions.AfterOpen = gotoAction;
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```
'Create a new document
Dim document As New PdfDocument()
'Add first page
Dim page As PdfPage = document.Pages.Add()
'Add second page
Dim secondPage As PdfPage = document.Pages.Add()
'Set the goto action
Dim gotoAction As New PdfGoToAction(secondPage)
'Set destination location
gotoAction.Destination = New PdfDestination(secondPage, New PointF(0, 100))
document.Actions.AfterOpen = gotoAction
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```
//Create a new document
PdfDocument document = new PdfDocument();
//Add first page
PdfPage page = document.Pages.Add();
//Add second page
PdfPage secondPage = document.Pages.Add();
//Set the goto action
PdfGoToAction gotoAction = new PdfGoToAction(secondPage);
//Set destination location
gotoAction.Destination = new PdfDestination(secondPage, new PointF(0, 100));
document.Actions.AfterOpen = gotoAction;
MemoryStream memoryStream = new MemoryStream();
//Save the document.

```

```
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(memoryStream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new document
PdfDocument document = new PdfDocument();
//Add first page
PdfPage page = document.Pages.Add();
//Add second page
PdfPage secondPage = document.Pages.Add();
//Set the goto action
PdfGoToAction gotoAction = new PdfGoToAction(secondPage);
//Set destination location
gotoAction.Destination = new PdfDestination(secondPage, new PointF(0, 100));
document.Actions.AfterOpen = gotoAction;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new document
PdfDocument document = new PdfDocument();
//Add first page
PdfPage page = document.Pages.Add();
//Add second page
PdfPage secondPage = document.Pages.Add();
//Set the goto action
PdfGoToAction gotoAction = new PdfGoToAction(secondPage);
//Set destination location
gotoAction.Destination = new PdfDestination(secondPage, new PointF(0, 100));
document.Actions.AfterOpen = gotoAction;
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
```



```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}

```

Launch action

A [PdfLaunchAction](#) allows execution of an external file. The code snippet below explains how to add a launch action.

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Create and add new launch Action to the document
PdfLaunchAction action = new PdfLaunchAction("../Data/logo.png");
document.Actions.AfterOpen = action;
//Save the document
document.Save("LaunchAction.pdf");
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As New PdfDocument()
'Create and add new launch Action to the document
Dim action As New PdfLaunchAction("../Data/logo.png")
document.Actions.AfterOpen = action
'Save the document
document.Save("LaunchAction.pdf")
document.Close(True)

```

UWP

```

//PDF supports launch action only in Windows Forms, WPF, ASP.NET and ASP.NET
MVC platforms.

```

ASP.NET CORE

```

//PDF supports launch action only in Windows Forms, WPF, ASP.NET and ASP.NET
MVC platforms.

```

XAMARIN

```

//PDF supports launch action only in Windows Forms, WPF, ASP.NET and ASP.NET
MVC platforms.

```

Named action

[PdfNamedAction](#) allows execution of predefined **PDF** actions.

The following predefined PDF actions are available:

- Go to next page
- Go to previous page
- Go to first page and
- Go to last page.

C#

```
//Create a new document
PdfDocument document = new PdfDocument();
document.Pages.Add();
document.Pages.Add();
//Create a named action
PdfNamedAction namedAction = new
PdfNamedAction(PdfActionDestination.LastPage);
//Add the named action
document.Actions.AfterOpen = namedAction;
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new document
Dim document As New PdfDocument()
document.Pages.Add()
document.Pages.Add()
'Create a named action
Dim namedAction As New PdfNamedAction(PdfActionDestination.LastPage)
'Add the named action
document.Actions.AfterOpen = namedAction
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a new document
PdfDocument document = new PdfDocument();
document.Pages.Add();
document.Pages.Add();
//Create a named action
PdfNamedAction namedAction = new
PdfNamedAction(PdfActionDestination.LastPage);
//Add the named action
document.Actions.AfterOpen = namedAction;
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
```

```
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples.
Save(memoryStream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new document
PdfDocument document = new PdfDocument();
document.Pages.Add();
document.Pages.Add();
//Create a named action
PdfNamedAction namedAction = new
PdfNamedAction(PdfActionDestination.LastPage);
//Add the named action
document.Actions.AfterOpen = namedAction;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new document
PdfDocument document = new PdfDocument();
document.Pages.Add();
document.Pages.Add();
//Create a named action
PdfNamedAction namedAction = new
PdfNamedAction(PdfActionDestination.LastPage);
//Add the named action
document.Actions.AfterOpen = namedAction;
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
```

```
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
    "application/pdf", memoryStream);
}
```

Submit action

[PdfSubmitAction](#) allows submission of data that is entered in the PDF form.

C#

```
//Create a PDF document
PdfDocument document = new PdfDocument();
//Add a new page
PdfPage page = document.Pages.Add();
// Create a Button field.
PdfButtonField submitButton = new PdfButtonField(page, "Submit data");
submitButton.Bounds = new RectangleF(100, 60, 50, 20);
submitButton.ToolTip = "Submit";
document.Form.Fields.Add(submitButton);
// Create a submit action. It submit the data of the form fields to the
mentioned URL
PdfSubmitAction submitAction = new
PdfSubmitAction("http://www.syncfusionforms.com/Submit.aspx");
submitAction.DataFormat = SubmitDataFormat.Html;
submitButton.Actions.GotFocus = submitAction;
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a PDF document
Dim document As New PdfDocument()
'Add a new page
Dim page As PdfPage = document.Pages.Add()
' Create a Button field.
Dim submitButton As New PdfButtonField(page, "Submit data")
submitButton.Bounds = New RectangleF(100, 60, 50, 20)
submitButton.ToolTip = "Submit"
document.Form.Fields.Add(submitButton)
' Create a submit action. It submit the data of the form fields to the
mentioned URL
Dim submitAction As New PdfSubmitAction("http://
www.example.com/Submit.aspx")
submitAction.DataFormat = SubmitDataFormat.Html
submitButton.Actions.GotFocus = submitAction
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a PDF document
PdfDocument document = new PdfDocument();
//Add a new page
```

```

PdfPage page = document.Pages.Add();
// Create a Button field.
PdfButtonField submitButton = new PdfButtonField(page, "Submit data");
submitButton.Bounds = new RectangleF(100, 60, 50, 20);
submitButton.ToolTip = "Submit";
document.Form.Fields.Add(submitButton);
// Create a submit action. It submit the data of the form fields to the
mentioned URL
PdfSubmitAction submitAction = new
PdfSubmitAction("http://www.syncfusionforms.com/Submit.aspx");
submitAction.DataFormat = SubmitDataFormat.Html;
submitButton.Actions.GotFocus = submitAction;
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(memoryStream, "Output.pdf");

```

ASP.NET CORE

```

//Create a PDF document
PdfDocument document = new PdfDocument();
//Add a new page
PdfPage page = document.Pages.Add();
// Create a Button field.
PdfButtonField submitButton = new PdfButtonField(page, "Submit data");
submitButton.Bounds = new RectangleF(100, 60, 50, 20);
submitButton.ToolTip = "Submit";
document.Form.Fields.Add(submitButton);
// Create a submit action. It submit the data of the form fields to the
mentioned URL
PdfSubmitAction submitAction = new
PdfSubmitAction("http://www.syncfusionforms.com/Submit.aspx");
submitAction.DataFormat = SubmitDataFormat.Html;
submitButton.Actions.GotFocus = submitAction;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a PDF document
PdfDocument document = new PdfDocument();

```

```
//Add a new page
PdfPage page = document.Pages.Add();
// Create a Button field.
PdfButtonField submitButton = new PdfButtonField(page, "Submit data");
submitButton.Bounds = new RectangleF(100, 60, 50, 20);
submitButton.ToolTip = "Submit";
document.Form.Fields.Add(submitButton);
// Create a submit action. It submit the data of the form fields to the
mentioned URL
PdfSubmitAction submitAction = new PdfSubmitAction("http://www.google.com");
submitAction.DataFormat = SubmitDataFormat.Html;
submitButton.Actions.GotFocus = submitAction;
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}
```

Reset action

A [PdfResetAction](#) allows execution of reset of all the form fields in the PDF document.

C#

```
//Create a PDF document
PdfDocument document = new PdfDocument();
//Add a new page
PdfPage page = document.Pages.Add();
// Create a Text box field.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
//Set properties to the textbox.
textBoxField.BorderColor = new PdfColor(Color.Gray);
textBoxField.BorderStyle = PdfBorderStyle.Beveled;
textBoxField.Bounds = new RectangleF(80, 0, 100, 20);
textBoxField.Text = "First Name";
//Add the form field to the document
document.Form.Fields.Add(textBoxField);
// Create a Button field.
PdfButtonField clearButton = new PdfButtonField(page, "Clear");
clearButton.Bounds = new RectangleF(100, 60, 50, 20);
clearButton.ToolTip = "Clear";
document.Form.Fields.Add(clearButton);
```

```
// Create an instance of reset action
PdfResetAction resetAction = new PdfResetAction();
clearButton.Actions.GotFocus = resetAction;
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a PDF document
Dim document As New PdfDocument()
'Add a new page
Dim page As PdfPage = document.Pages.Add()
' Create a Text box field.
Dim textBoxField As New PdfTextBoxField(page, "FirstName")
'Set properties to the textbox.
textBoxField.BorderColor = New PdfColor(Color.Gray)
textBoxField.BorderStyle = PdfBorderStyle.Beveled
textBoxField.Bounds = New RectangleF(80, 0, 100, 20)
textBoxField.Text = "First Name"
'Add the form field to the document
document.Form.Fields.Add(textBoxField)
' Create a Button field.
Dim clearButton As New PdfButtonField(page, "Clear")
clearButton.Bounds = New RectangleF(100, 60, 50, 20)
clearButton.ToolTip = "Clear"
document.Form.Fields.Add(clearButton)
' Create an instance of reset action
Dim resetAction As New PdfResetAction()
clearButton.Actions.GotFocus = resetAction
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a PDF document
PdfDocument document = new PdfDocument();
//Add a new page
PdfPage page = document.Pages.Add();
// Create a Text box field.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
//Set properties to the textbox.
textBoxField.BorderColor = new PdfColor(128, 128, 128);
textBoxField.BorderStyle = PdfBorderStyle.Beveled;
textBoxField.Bounds = new RectangleF(80, 0, 100, 20);
textBoxField.Text = "First Name";
//Add the form field to the document
document.Form.Fields.Add(textBoxField);
// Create a Button field.
PdfButtonField clearButton = new PdfButtonField(page, "Clear");
clearButton.Bounds = new RectangleF(100, 60, 50, 20);
clearButton.ToolTip = "Clear";
document.Form.Fields.Add(clearButton);
// Create an instance of reset action
PdfResetAction resetAction = new PdfResetAction();
```

```
clearButton.Actions.GotFocus = resetAction;
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(memoryStream, "Output.pdf");
```

ASP.NET CORE

```
//Create a PDF document
PdfDocument document = new PdfDocument();
//Add a new page
PdfPage page = document.Pages.Add();
// Create a Text box field.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
//Set properties to the textbox.
textBoxField.BorderColor = new PdfColor(Color.Gray);
textBoxField.BorderStyle = PdfBorderStyle.Beveled;
textBoxField.Bounds = new RectangleF(80, 0, 100, 20);
textBoxField.Text = "First Name";
//Add the form field to the document
document.Form.Fields.Add(textBoxField);
// Create a Button field.
PdfButtonField clearButton = new PdfButtonField(page, "Clear");
clearButton.Bounds = new RectangleF(100, 60, 50, 20);
clearButton.ToolTip = "Clear";
document.Form.Fields.Add(clearButton);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a PDF document
PdfDocument document = new PdfDocument();
//Add a new page
PdfPage page = document.Pages.Add();
// Create a Text box field.
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "FirstName");
//Set properties to the textbox.
textBoxField.BorderColor = new PdfColor(Syncfusion.Drawing.Color.Gray);
textBoxField.BorderStyle = PdfBorderStyle.Beveled;
textBoxField.Bounds = new RectangleF(80, 0, 100, 20);
```



```

textBoxField.Text = "First Name";
//Add the form field to the document
document.Form.Fields.Add(textBoxField);
// Create a Button field.
PdfButtonField clearButton = new PdfButtonField(page, "Clear");
clearButton.Bounds = new RectangleF(100, 60, 50, 20);
clearButton.ToolTip = "Clear";
document.Form.Fields.Add(clearButton);
// Create an instance of reset action
PdfResetAction resetAction = new PdfResetAction();
clearButton.Actions.GotFocus = resetAction;
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}

```

Adding an action to the form field

Essential PDF provides support to add various actions to the form fields.

[PdfFieldActions](#) class is used to create form field actions.

The following code example illustrates this.

C#

```

//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Create a new PdfButtonField
PdfButtonField submitButton = new PdfButtonField(page, "submitButton");
submitButton.Bounds = new RectangleF(25, 160, 100, 20);
submitButton.Text = "Apply";
submitButton.BackColor = new PdfColor(181, 191, 203);
//Create a new PdfJavaScriptAction
PdfJavaScriptAction scriptAction = new PdfJavaScriptAction("app.alert(\"You
are looking at Form field action of PDF \")");
//Set the scriptAction to submitButton
submitButton.Actions.MouseDown = scriptAction;
//Add the submit button to the new document.
document.Form.Fields.Add(submitButton);

```

```
//Save document to disk.
document.Save("fieldAction.pdf");
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Create a new PdfButtonField
Dim submitButton As New PdfButtonField(page, "submitButton")
submitButton.Bounds = New RectangleF(25, 160, 100, 20)
submitButton.Text = "Apply"
submitButton.BackColor = New PdfColor(181, 191, 203)
'Create a new PdfJavaScriptAction
Dim scriptAction As New PdfJavaScriptAction("app.alert(\"\"You are looking at
Form field action of PDF \"\")")
'Set the scriptAction to submitButton
submitButton.Actions.MouseDown = scriptAction
'Add the submit button to the new document.
document.Form.Fields.Add(submitButton)
'Save document to disk.
document.Save("fieldAction.pdf")
document.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Create a new PdfButtonField
PdfButtonField submitButton = new PdfButtonField(page, "submitButton");
submitButton.Bounds = new RectangleF(25, 160, 100, 20);
submitButton.Text = "Apply";
submitButton.BackColor = new PdfColor(181, 191, 203);
//Create a new PdfJavaScriptAction
PdfJavaScriptAction scriptAction = new PdfJavaScriptAction("app.alert(\"\"You
are looking at Form field action of PDF \"\")");
//Set the scriptAction to submitButton
submitButton.Actions.MouseDown = scriptAction;
//Add the submit button to the new document.
document.Form.Fields.Add(submitButton);
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(memoryStream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Create a new PdfButtonField
PdfButtonField submitButton = new PdfButtonField(page, "submitButton");
submitButton.Bounds = new RectangleF(25, 160, 100, 20);
submitButton.Text = "Apply";
submitButton.BackColor = new PdfColor(181, 191, 203);
//Create a new PdfJavaScriptAction
PdfJavaScriptAction scriptAction = new PdfJavaScriptAction("app.alert(\"You
are looking at Form field action of PD
//Set the scriptAction to submitButton
submitButton.Actions.MouseDown = scriptAction;
//Add the submit button to the new document.
document.Form.Fields.Add(submitButton);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Create a new PdfButtonField
PdfButtonField submitButton = new PdfButtonField(page, "submitButton");
submitButton.Bounds = new RectangleF(25, 160, 100, 20);
submitButton.Text = "Apply";
submitButton.BackColor = new PdfColor(181, 191, 203);
//Create a new PdfJavaScriptAction
PdfJavaScriptAction scriptAction = new PdfJavaScriptAction("app.alert(\"You
are looking at Form field action of PDF \")");
//Set the scriptAction to submitButton
submitButton.Actions.MouseDown = scriptAction;
//Add the submit button to the new document.
document.Form.Fields.Add(submitButton);
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream into pdf file
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android
//and iOS platforms. Please refer PDF/Xamarin section for respective code
//samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}
```

Adding an action to the bookmarks

Essential PDF provides support to add the various actions to the [Bookmarks](#). The code snippet below shows how to add an URI action to bookmark.

C#

```
//Create a new document.
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create document bookmarks.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Set the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Color.Red;
//Create a Uri action
PdfUriAction uriAction = new PdfUriAction("http://www.google.com");
//Set the Uri action
bookmark.Action = uriAction;
//Save and close the PDF document.
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new document.
Dim document As New PdfDocument()
'Add a page.
Dim page As PdfPage = document.Pages.Add()
'Create document bookmarks.
Dim bookmark As PdfBookmark = document.Bookmarks.Add("Page 1")
'Set the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold
bookmark.Color = Color.Red
'Create a Uri action
Dim uriAction As New PdfUriAction("http://www.google.com")
'Set the Uri action
bookmark.Action = uriAction
'Save and close the PDF document.
document.Save("Output.pdf")
```

```
document.Close(True)
```

UWP

```
//Create a new document.
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create document bookmarks.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Set the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Color.FromArgb(0,255,0,0);
//Create a Uri action
PdfUriAction uriAction = new PdfUriAction("http://www.google.com");
//Set the Uri action
bookmark.Action = uriAction;
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(memoryStream, "fieldAction.pdf");
```

ASP.NET CORE

```
//Create a new document.
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create document bookmarks.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Set the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Color.Red;
//Create a Uri action
PdfUriAction uriAction = new PdfUriAction("http://www.google.com");
//Set the Uri action
bookmark.Action = uriAction;
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```

//Create a new document.
PdfDocument document = new PdfDocument();
//Add a page.
PdfPage page = document.Pages.Add();
//Create document bookmarks.
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
//Set the text style and color.
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Color = Syncfusion.Drawing.Color.Red;
//Create a Uri action
PdfUriAction uriAction = new PdfUriAction("http://www.google.com");
//Set the Uri action
bookmark.Action = uriAction;
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}

```

Note: The action assigned to the bookmark works only when destination of bookmark is not set.

Working with Watermarks

Essential PDF provides you support to add watermark in the PDF document using PdfGraphics.

Adding text watermark in PDF document

Essential PDF allows you to draw the text watermark in PDF document using graphics elements.

The below code illustrates how to draw the text watermark in new PDF document:

C#

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
PdfGraphics graphics = pdfPage.Graphics;
//set the font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//watermark text.
PdfGraphicsState state = graphics.Save();

```

```
graphics.SetTransparency(0.25f);
graphics.RotateTransform(-40);
graphics.DrawString("Imported using Essential PDF", font, PdfPens.Red,
PdfBrushes.Red, new PointF(-150, 450));
//Save and close the document.
pdfDocument.Save("watermark.pdf");
pdfDocument.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim pdfDocument As New PdfDocument()
'Add a page to the PDF document.
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
Dim graphics As PdfGraphics = pdfPage.Graphics
'set the font
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
' watermark text.
Dim state As PdfGraphicsState = graphics.Save()
graphics.SetTransparency(0.25F)
graphics.RotateTransform(-40)
graphics.DrawString("Imported using Essential PDF", font, PdfPens.Red,
PdfBrushes.Red, New PointF(-150, 450))
'Save and close the document.
pdfDocument.Save("watermark.pdf")
pdfDocument.Close(True)
```

UWP

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
PdfGraphics graphics = pdfPage.Graphics;
//set the font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//watermark text.
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.RotateTransform(-40);
graphics.DrawString("Imported using Essential PDF", font, PdfPens.Red,
PdfBrushes.Red, new PointF(-150, 450));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
//Close the document
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
```

```
//Add a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
PdfGraphics graphics = pdfPage.Graphics;
//set the font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//watermark text.
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.RotateTransform(-40);
graphics.DrawString("Imported using Essential PDF", font, PdfPens.Red,
PdfBrushes.Red, new PointF(-150, 450));
//Save the document into stream.
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Close the document.
pdfDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "watermark.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

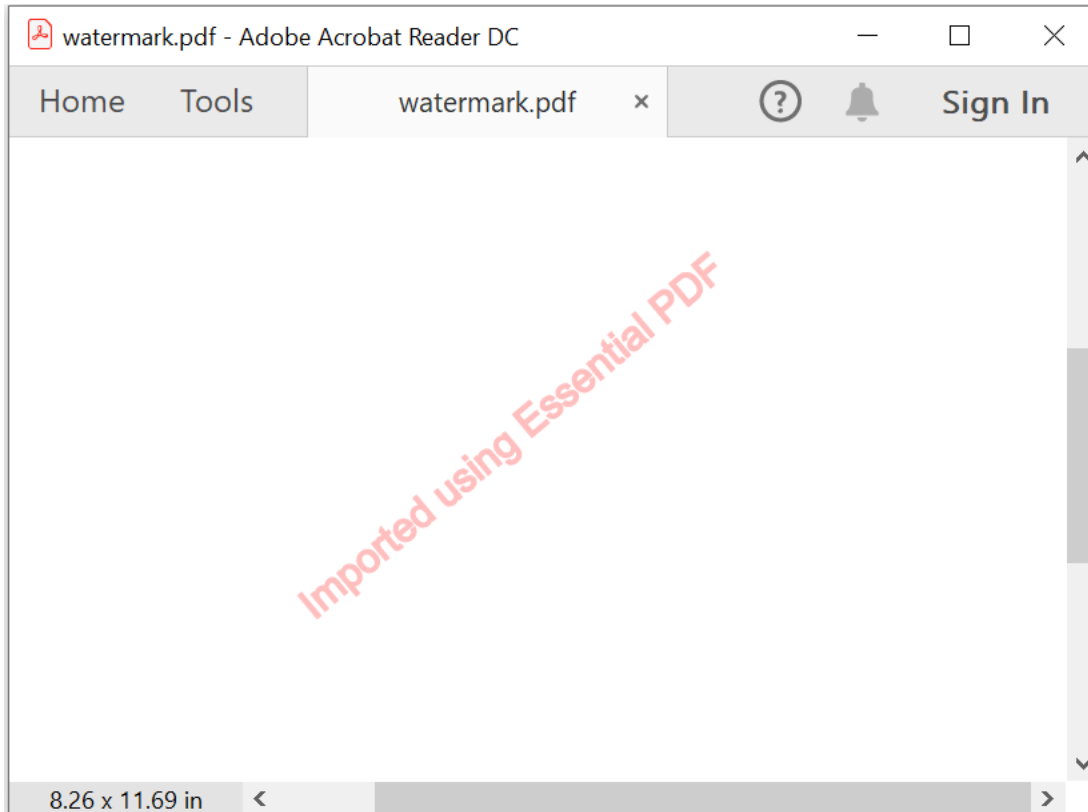
XAMARIN

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
PdfGraphics graphics = pdfPage.Graphics;
//set the font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
//watermark text.
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.RotateTransform(-40);
graphics.DrawString("Imported using Essential PDF", font, PdfPens.Red,
PdfBrushes.Red, new PointF(-150, 450));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Closes the document
pdfDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
```



```
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
    "application/pdf", stream);
}
```

The following screenshot shows the output of adding text watermark to PDF document.



The below code illustrates how to draw the text watermark in existing PDF document:

C#

```
//Load the document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
PdfPageBase loadedPage = loadedDocument.Pages[0];
PdfGraphics graphics = loadedPage.Graphics;
//set the font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
// watermark text.
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.RotateTransform(-40);
graphics.DrawString("Imported using Essential PDF", font, PdfPens.Red,
PdfBrushes.Red, new PointF(-150, 450));
//Save and close the document.
loadedDocument.Save("watermark.pdf");
loadedDocument.Close(true);
```

VB.NET

```

'Load the document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
Dim loadedPage As PdfPageBase = loadedDocument.Pages(0)
Dim graphics As PdfGraphics = loadedPage.Graphics
'set the font
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
' watermark text.
Dim state As PdfGraphicsState = graphics.Save()
graphics.SetTransparency(0.25F)
graphics.RotateTransform(-40)
graphics.DrawString("Imported using Essential PDF", font, PdfPens.Red,
PdfBrushes.Red, New PointF(-150, 450))
'Save and close the document.
loadedDocument.Save("watermark.pdf")
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
PdfPageBase loadedPage = loadedDocument.Pages[0];
PdfGraphics graphics = loadedPage.Graphics;
//set the font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
// watermark text.
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.RotateTransform(-40);
graphics.DrawString("Imported using Essential PDF", font, PdfPens.Red,
PdfBrushes.Red, new PointF(-150, 450));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
PdfPageBase loadedPage = loadedDocument.Pages[0];
PdfGraphics graphics = loadedPage.Graphics;
//set the font

```

```

PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
// watermark text.
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.RotateTransform(-40);
graphics.DrawString("Imported using Essential PDF", font, PdfPens.Red,
PdfBrushes.Red, new PointF(-150,
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "watermark.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

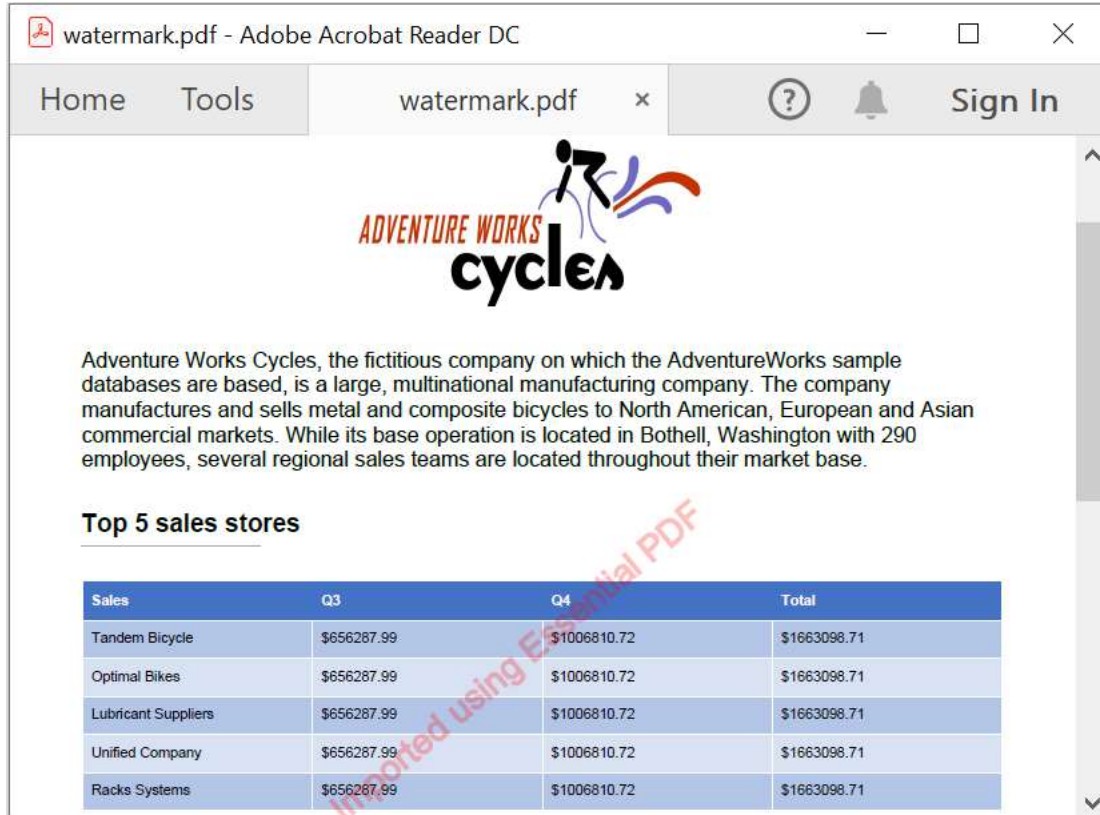
```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
PdfPageBase loadedPage = loadedDocument.Pages[0];
PdfGraphics graphics = loadedPage.Graphics;
//set the font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
// watermark text.
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.RotateTransform(-40);
graphics.DrawString("Imported using Essential PDF", font, PdfPens.Red,
PdfBrushes.Red, new PointF(-150, 450));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Closes the document
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{

```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

The following screenshot shows the output of adding text watermark to an existing PDF document.



Adding image watermark in PDF document

To add the image watermark in PDF document, you can draw the image with transparency in PdfGraphics.

The below code illustrates how to draw the image watermark in new PDF document:

C#

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
PdfGraphics graphics = pdfPage.Graphics;
//Image watermark.
PdfImage image = new PdfBitmap("GifImage.gif");
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.DrawImage(image, new PointF(0, 0), pdfPage.Graphics.ClientSize);
//Save and close the document.
pdfDocument.Save("watermark.pdf");
pdfDocument.Close(true);
```

VB.NET

```

'Create a new PDF document.
Dim pdfDocument As New PdfDocument()
'Add a page to the PDF document.
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
Dim graphics As PdfGraphics = pdfPage.Graphics
'Image watermark.
Dim image As PdfImage = New PdfBitmap("GifImage.gif")
Dim state As PdfGraphicsState = graphics.Save()
graphics.SetTransparency(0.25F)
graphics.DrawImage(image, New PointF(0, 0), pdfPage.Graphics.ClientSize)
'Save and close the document.
pdfDocument.Save("watermark.pdf")
pdfDocument.Close(True)

```

UWP

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
PdfGraphics graphics = pdfPage.Graphics;
//Load the image as stream.
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.GifImage.gif");
//Image watermark.
PdfImage image = new PdfBitmap(imageStream);
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.DrawImage(image, new PointF(0, 0), pdfPage.Graphics.ClientSize);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
//Close the document
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
PdfGraphics graphics = pdfPage.Graphics;
//Load the image as stream.
FileStream imageStream = new FileStream("GifImage.gif", FileMode.Open,
FileAccess.Read);
//Image watermark.
PdfImage image = new PdfBitmap(imageStream);
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.DrawImage(image, new PointF(0, 0), pdfPage.Graphics.ClientSize);

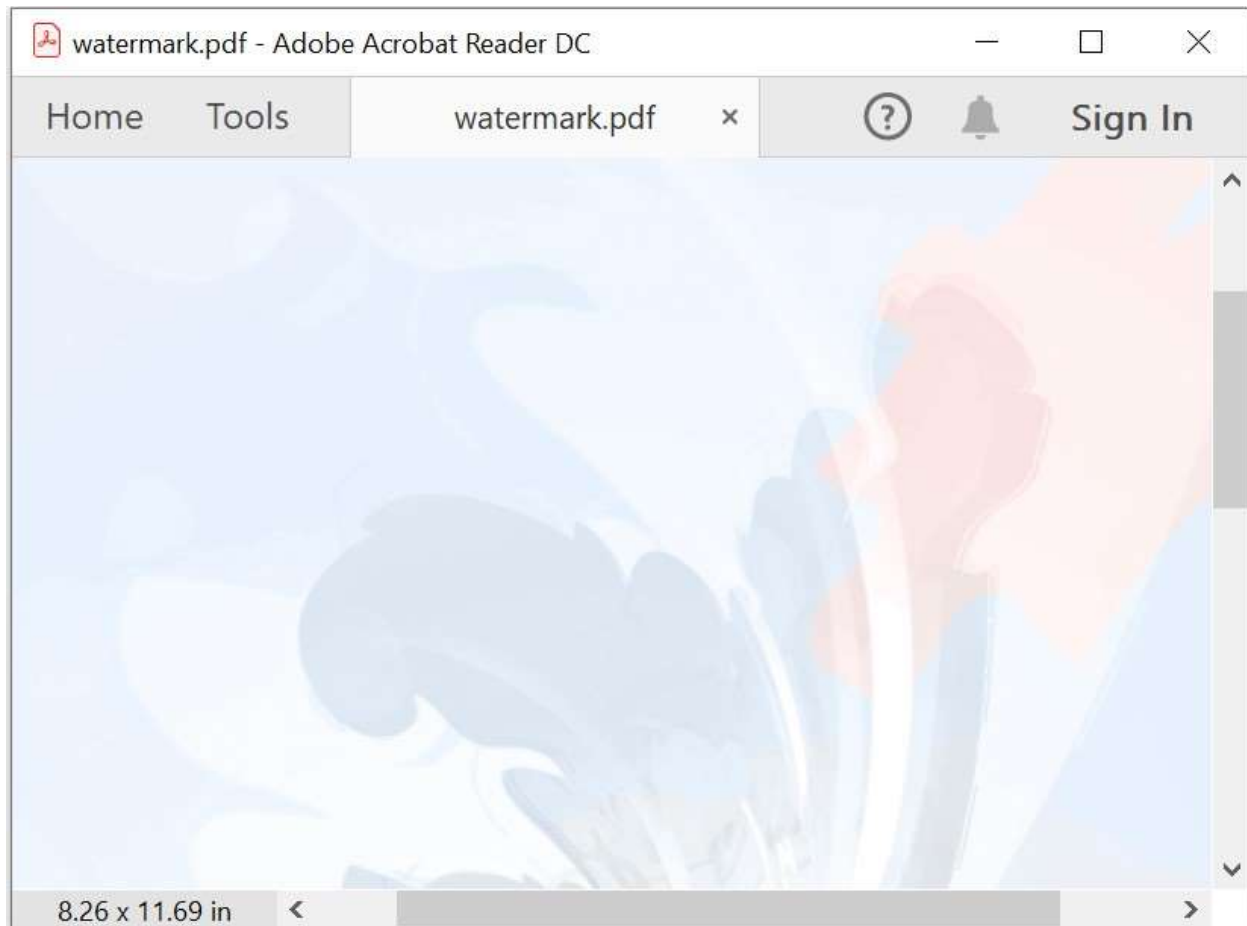
```

```
//Save the document into stream.
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Close the document.
pdfDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "watermark.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
PdfGraphics graphics = pdfPage.Graphics;
//Load the image as stream.
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
GifImage.gif");
PdfImage image = new PdfBitmap(imageStream);
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.DrawImage(image, new PointF(0, 0), pdfPage.Graphics.ClientSize);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Closes the document
pdfDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

The following screenshot shows the output of adding image watermark to PDF document.



The below code illustrates how to draw the image watermark in existing PDF document.

C#

```
//Load the document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
PdfPageBase loadedPage = loadedDocument.Pages[0];
PdfGraphics graphics = loadedPage.Graphics;
//Image watermark.
PdfImage image = new PdfBitmap("GifImage.gif");
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.DrawImage(image, new PointF(0, 0), loadedPage.Graphics.ClientSize);
//Save and close the document.
loadedDocument.Save("watermark.pdf");
loadedDocument.Close(true);
```

VB.NET

```
'Load the document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
Dim loadedPage As PdfPageBase = loadedDocument.Pages(0)
Dim graphics As PdfGraphics = loadedPage.Graphics
'Image watermark.
Dim image As PdfImage = New PdfBitmap("GifImage.gif")
Dim state As PdfGraphicsState = graphics.Save()
```

```
graphics.SetTransparency(0.25f)
graphics.DrawImage(image, new PointF(0, 0), loadedPage.Graphics.ClientSize)
'Save and close the document.'
loadedDocument.Save("watermark.pdf")
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
PdfPageBase loadedPage = loadedDocument.Pages[0];
PdfGraphics graphics = loadedPage.Graphics;
//Load the image as stream
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.Data.GifImage.gif");
//Image watermark.
PdfImage image = new PdfBitmap(imageStream);
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.DrawImage(image, new PointF(0, 0), loadedPage.Graphics.ClientSize);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples
Save(stream, "output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open, FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
PdfPageBase loadedPage = loadedDocument.Pages[0];
PdfGraphics graphics = loadedPage.Graphics;
//Load the image file as stream
FileStream imageStream = new FileStream("GifImage.gif", FileMode.Open, FileAccess.Read);
PdfImage image = new PdfBitmap(imageStream);
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.DrawImage(image, new PointF(0, 0), loadedPage.Graphics.ClientSize);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
```



```

stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "watermark.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

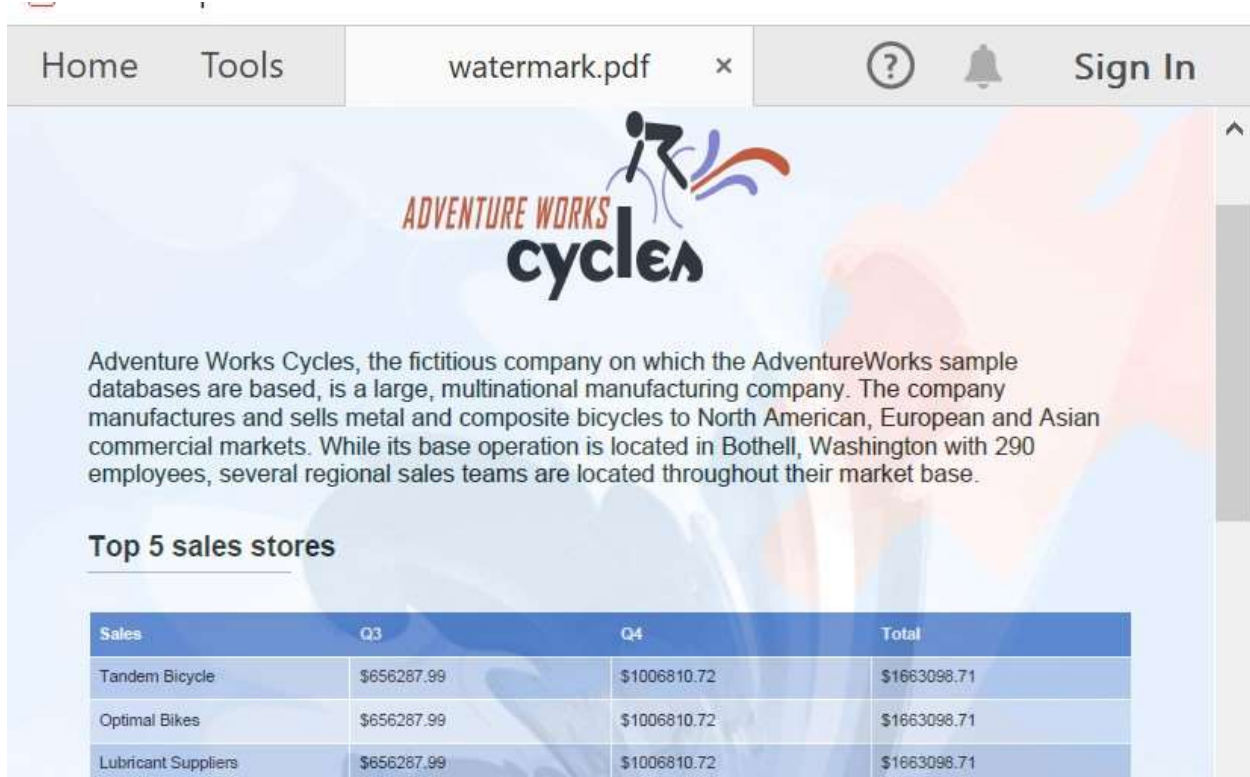
XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
PdfPageBase loadedPage = loadedDocument.Pages[0];
PdfGraphics graphics = loadedPage.Graphics;
//Load the image as stream.
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
GifImage.gif");
PdfImage image = new PdfBitmap(imageStream);
PdfGraphicsState state = graphics.Save();
graphics.SetTransparency(0.25f);
graphics.DrawImage(image, new PointF(0, 0), loadedPage.Graphics.ClientSize);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Closes the document
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following screenshot shows the output of adding image watermark to an existing PDF document.



Working with Portfolio

PDF Portfolios allows the user to bring together content from a variety of sources, including documents, drawings, images, e-mail, spreadsheets and web pages. Essential PDF allows the user to create portfolios and also to extract the files from them.

Creating a PDF portfolio

You can create a portfolio using [PdfPortfolioInformation](#) class and attach a variety of documents. The following code snippet illustrates this.

C#

```
// Create a new instance of PdfDocument class.
PdfDocument document = new PdfDocument();
//Create a new portfolio
document.PortfolioInformation = new PdfPortfolioInformation();
//set the view mode of the portfolio
document.PortfolioInformation.ViewMode = PdfPortfolioViewMode.Tile;
//Create the attachment
PdfAttachment pdfFile = new
PdfAttachment("../Data/CorporateBrochure.pdf");
pdfFile.FileName = "CorporateBrochure.pdf";
//Set the startup document to view
document.PortfolioInformation.StartupDocument = pdfFile;
//Add the attachment to the document
document.Attachments.Add(pdfFile);
// Save and close the document.
document.Save("Sample.pdf");
document.Close(true);
```

VB.NET

```

' Create a new instance of PdfDocument class.
Dim document As New PdfDocument()
'Create a new portfolio
document.PortfolioInformation = New PdfPortfolioInformation()
'Set the view mode of the portfolio
document.PortfolioInformation.ViewMode = PdfPortfolioViewMode.Tile
'Create the attachment
Dim pdfFile As New PdfAttachment("../Data/CorporateBrochure.pdf")
pdfFile.FileName = "CorporateBrochure.pdf"
'Set the startup document to view
document.PortfolioInformation.StartupDocument = pdfFile
'Add the attachment to the document
document.Attachments.Add(pdfFile)
'Save and close the document.
document.Save("Sample.pdf")
document.Close(True)

```

UWP

```

// Create a new instance of PdfDocument class.
PdfDocument document = new PdfDocument();
//Create a new portfolio
document.PortfolioInformation = new PdfPortfolioInformation();
//set the view mode of the portfolio
document.PortfolioInformation.ViewMode = PdfPortfolioViewMode.Tile;
//Create the attachment
Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Sample.pdf");
PdfAttachment pdfFile = new PdfAttachment("Sample.pdf", pdfStream);
pdfFile.FileName = "Sample.pdf";
//Set the startup document to view
document.PortfolioInformation.StartupDocument = pdfFile;
//Add the attachment to the document
document.Attachments.Add(pdfFile);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

// Create a new instance of PdfDocument class.
PdfDocument document = new PdfDocument();
//Create a new portfolio
document.PortfolioInformation = new PdfPortfolioInformation();
//set the view mode of the portfolio
document.PortfolioInformation.ViewMode = PdfPortfolioViewMode.Tile;
//Create the attachment

```

```

FileStream pdfStream = new FileStream("CorporateBrochure.pdf",
    FileMode.Open, FileAccess.Read);
PdfAttachment pdfFile = new PdfAttachment("CorporateBrochure.pdf",
    pdfStream);
pdfFile.FileName = "CorporateBrochure.pdf";
//Set the startup document to view
document.PortfolioInformation.StartupDocument = pdfFile;
//Add the attachment to the document
document.Attachments.Add(pdfFile);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Sample.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

// Create a new instance of PdfDocument class.
PdfDocument document = new PdfDocument();
//Create a new portfolio
document.PortfolioInformation = new PdfPortfolioInformation();
//set the view mode of the portfolio
document.PortfolioInformation.ViewMode = PdfPortfolioViewMode.Tile;
//Create the attachment
Stream docStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
    Sample.pdf");
PdfAttachment pdfFile = new PdfAttachment("Sample.pdf", docStream);
pdfFile.FileName = "Sample.pdf";
//Set the startup document to view
document.PortfolioInformation.StartupDocument = pdfFile;
//Add the attachment to the document
document.Attachments.Add(pdfFile);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
    "application/pdf", stream);
}

```

```
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Extracting file from PDF Portfolio

Essential PDF also provides support for extracting the files from the PDF Portfolio and saving it to the disk. The following code snippet shows the steps to extract files from PDF Portfolio.

C#

```
//Load the PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Sample.pdf");
//Iterate the attachments
foreach (PdfAttachment attachment in document.Attachments)
{
//Extract the attachment and save to the disk
FileStream s = new FileStream(attachment.FileName, FileMode.Create);
s.Write(attachment.Data, 0, attachment.Data.Length);
s.Dispose();
}
//Save and close the document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Load the PDF document
Dim document As New PdfLoadedDocument("Sample.pdf")
'Iterate the attachments
For Each attachment As PdfAttachment In document.Attachments
'Extracting the attachment and saving into the local disk
Dim s As New FileStream(attachment.FileName, FileMode.Create)
s.Write(attachment.Data, 0, attachment.Data.Length)
s.Dispose()
Next
'Save and close the document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//PDF supports extracting file from PDF Portfolio only in Windows Forms,
WPF, ASP.NET and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Sample.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Iterate the attachments
foreach (PdfAttachment attachment in document.Attachments)
```

```
{
//Extract the attachment and save to the disk
FileStream s = new FileStream(attachment.FileName, FileMode.Create);
s.Write(attachment.Data, 0, attachment.Data.Length);
s.Dispose();
}
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//PDF supports extracting file from PDF Portfolio only in Windows Forms,
WPF, ASP.NET and ASP.NET MVC platforms.
```

Removing files from PDF Portfolio

You can also remove the files from PDF Portfolio using following code.

C#

```
//Load the PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Sample.pdf");
//Remove the file from the Portfolio
document.Attachments.RemoveAt(1);
//Save and close the document
document.Save("Output.pdf");
document.Close();
```

VB.NET

```
'Load the PDF document
Dim document As New PdfLoadedDocument("Sample.pdf")
'Remove the file from the Portfolio
document.Attachments.RemoveAt(1)
'Save and close the document
document.Save("Output.pdf")
document.Close()
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
```

```

//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await document.OpenAsync(file);
//Remove the file from the Portfolio
document.Attachments.RemoveAt(1);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Sample.pdf", FileMode.Open, FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Remove the file from the Portfolio
document.Attachments.RemoveAt(0);
//Save and close the document
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

// Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.Sample.pdf");
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Remove the file from the Portfolio
document.Attachments.RemoveAt(0);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Closes the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer PDF/Xamarin section for respective code samples

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Working with Layers

Layers, also known as Option Content refers to sections of content in a PDF document that can be selectively viewed or hidden by document authors or consumers. This capability is useful in items such as CAD drawings, layered artwork, maps, and multi-language documents.

Essential PDF provides support to create, add and merge the layers into PDF document.

Adding Layers in a PDF document

Essential PDF allows the users to create a layer in a PDF page using [PdfPageLayer](#) class. The below code snippet illustrates how to add the multiple layers in a new PDF document.

C#

```

//Create PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
//Add the first layer.
PdfPageLayer layer = page.Layers.Add("Layer1");
PdfGraphics graphics = layer.Graphics;
graphics.TranslateTransform(100, 60);
//Draw arc.
PdfPen pen = new PdfPen(System.Drawing.Color.Red, 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add another layer on the page.
PdfPageLayer layer2 = page.Layers.Add("Layer2");
graphics = layer2.Graphics;
graphics.TranslateTransform(100, 180);
//Draw ellipse.
graphics.DrawEllipse(pen, bounds);
//Save the document.
document.Save("Sample.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Create PDF document.
Dim document As New PdfDocument()
Dim page As PdfPage = document.Pages.Add()
'Add the first layer.
Dim layer As PdfPageLayer = page.Layers.Add("Layer1")
Dim graphics As PdfGraphics = layer.Graphics

```



```

graphics.TranslateTransform(100, 60)
'Draw arc.
Dim pen As New PdfPen(System.Drawing.Color.Red, 50)
Dim bounds As New RectangleF(0, 0, 50, 50)
graphics.DrawArc(pen, bounds, 360, 360)
'Add another layer on the page.
Dim layer2 As PdfPageLayer = page.Layers.Add("Layer2")
graphics = layer2.Graphics
graphics.TranslateTransform(100, 180)
'Draw ellipse.
graphics.DrawEllipse(pen, bounds)
'Save the document.
document.Save("Sample.pdf")
'Close the document
document.Close(True)

```

UWP

```

//Create PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
//Add the first layer.
PdfPageLayer layer = page.Layers.Add("Layer1");
PdfGraphics graphics = layer.Graphics;
graphics.TranslateTransform(100, 60);
//Draw arc.
PdfPen pen = new PdfPen(new PdfColor(255, 0, 0), 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add another layer on the page.
PdfPageLayer layer2 = page.Layers.Add("Layer2");
graphics = layer2.Graphics;
graphics.TranslateTransform(100, 180);
//Draw ellipse.
graphics.DrawEllipse(pen, bounds);
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(memoryStream, "Sample.pdf");

```

ASP.NET CORE

```

//Create PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
//Add the first layer.
PdfPageLayer layer = page.Layers.Add("Layer1");
PdfGraphics graphics = layer.Graphics;
graphics.TranslateTransform(100, 60);
//Draw arc.
PdfPen pen = new PdfPen(Syncfusion.Drawing.Color.Red, 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);

```

```

graphics.DrawArc(pen, bounds, 360, 360);
//Add another layer on the page.
PdfPageLayer layer2 = page.Layers.Add("Layer2");
graphics = layer2.Graphics;
graphics.TranslateTransform(100, 180);
//Draw ellipse.
graphics.DrawEllipse(pen, bounds);
//Save and close the document
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create PDF document.
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
//Add the first layer.
PdfPageLayer layer = page.Layers.Add("Layer1");
PdfGraphics graphics = layer.Graphics;
graphics.TranslateTransform(100, 60);
//Draw arc.
PdfPen pen = new PdfPen(Syncfusion.Drawing.Color.Red, 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add another layer on the page.
PdfPageLayer layer2 = page.Layers.Add("Layer2");
graphics = layer2.Graphics;
graphics.TranslateTransform(100, 180);
//Draw ellipse.
graphics.DrawEllipse(pen, bounds);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pdf",
"application/pdf", stream);
}
else

```

```
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pdf",
"application/pdf", stream);
}
```

The below code illustrates how to add the multiple layers in an existing PDF document.

C#

```
//Load the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Add the first layer.
PdfPageLayer layer = loadedPage.Layers.Add("Layer1");
PdfGraphics graphics = layer.Graphics;
graphics.TranslateTransform(100, 60);
//Draw arc.
PdfPen pen = new PdfPen(System.Drawing.Color.Gray, 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add another layer on the page.
PdfPageLayer layer2 = loadedPage.Layers.Add("Layer2");
graphics = layer2.Graphics;
graphics.TranslateTransform(100, 180);
//Draw ellipse.
graphics.DrawEllipse(pen, bounds);
//Save the document.
loadedDocument.Save("Output.pdf");
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Add the first layer.
Dim layer As PdfPageLayer = loadedPage.Layers.Add("Layer1")
Dim graphics As PdfGraphics = layer.Graphics
graphics.TranslateTransform(100, 60)
'Draw arc.
Dim pen As New PdfPen(System.Drawing.Color.Gray, 50)
Dim bounds As New RectangleF(0, 0, 50, 50)
graphics.DrawArc(pen, bounds, 360, 360)
'Add another layer on the page.
Dim layer2 As PdfPageLayer = loadedPage.Layers.Add("Layer2")
graphics = layer2.Graphics
graphics.TranslateTransform(100, 180)
'Draw ellipse.
graphics.DrawEllipse(pen, bounds)
'Save the document.
loadedDocument.Save("Output.pdf")
'Close the document
loadedDocument.Close(True)
```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Add the first layer.
PdfPageLayer layer = loadedPage.Layers.Add("Layer1");
PdfGraphics graphics = layer.Graphics;
graphics.TranslateTransform(100, 60);
//Draw arc.
PdfPen pen = new PdfPen(new PdfColor(255, 0, 0), 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add another layer on the page.
PdfPageLayer layer2 = loadedPage.Layers.Add("Layer2");
graphics = layer2.Graphics;
graphics.TranslateTransform(100, 180);
//Draw ellipse.
graphics.DrawEllipse(pen, bounds);
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await loadedDocument.SaveAsync(memoryStream);
//Close the documents.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples.
Save(memoryStream, "Sample.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open, FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Add the first layer.
PdfPageLayer layer = loadedPage.Layers.Add("Layer1");
PdfGraphics graphics = layer.Graphics;
graphics.TranslateTransform(100, 60);
//Draw arc.
PdfPen pen = new PdfPen(Syncfusion.Drawing.Color.Gray, 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add another layer on the page.
PdfPageLayer layer2 = loadedPage.Layers.Add("Layer2");
graphics = layer2.Graphics;
graphics.TranslateTransform(100, 180);
//Draw ellipse.

```

```

graphics.DrawEllipse(pen, bounds);
//Save and close the document
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Add the first layer.
PdfPageLayer layer = loadedPage.Layers.Add("Layer1");
PdfGraphics graphics = layer.Graphics;
graphics.TranslateTransform(100, 60);
//Draw arc.
PdfPen pen = new PdfPen(Syncfusion.Drawing.Color.Gray, 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add another layer on the page.
PdfPageLayer layer2 = loadedPage.Layers.Add("Layer2");
graphics = layer2.Graphics;
graphics.TranslateTransform(100, 180);
//Draw ellipse.
graphics.DrawEllipse(pen, bounds);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pdf",
"application/pdf", stream);
}

```

```
}

```

Adding annotation to layer

Essential PDF allows the users to add [Annotation](#) to layers in the PDF document. Refer to the following code snippet.

C#

```
//Create new PDF document
PdfDocument document = new PdfDocument();
//Add page
PdfPage page = document.Pages.Add();
//Add the layer
PdfLayer layer = document.Layers.Add("Layer");
//Create graphics for layer
PdfGraphics graphics = layer.CreateGraphics(page);
//Draw ellipse
graphics.DrawEllipse(PdfPens.Red, new RectangleF(50, 50, 40, 40));
//Create square annotation
PdfSquareAnnotation annotation = new PdfSquareAnnotation(new RectangleF(200,
260, 50, 50), "Square annotation");
annotation.Color = new PdfColor(Color.Red);
//Set layer to annotation
annotation.Layer = layer;
//Add annotation to the created page
page.Annotations.Add(annotation);
//Save the document
document.Save("Output.pdf");
//Close the document
document.Close(true);
```

VB.NET

```
'Create new PDF document
Dim document As New PdfDocument()
'Add page
Dim page As PdfPage = document.Pages.Add()
'Add the layer
Dim Layer As PdfLayer = document.Layers.Add("Layer")
'Create graphics for layer
Dim graphics As PdfGraphics = Layer.CreateGraphics(page)
'Draw ellipse
graphics.DrawEllipse(PdfPens.Red, New RectangleF(50, 50, 40, 40))
'Create square annotation
Dim annotation As New PdfSquareAnnotation(New RectangleF(200, 260, 50, 50),
"Square annotation")
annotation.Color = New PdfColor(Color.Red)
'Set layer to annotation
annotation.Layer = Layer
'Add annotation to the created page
page.Annotations.Add(annotation)
'Save the document
document.Save("Output.pdf")
'Close the document
document.Close(True)
```

UWP

```

//Create new PDF document
PdfDocument document = new PdfDocument();
//Add page
PdfPage page = document.Pages.Add();
//Add the layer
PdfLayer layer = document.Layers.Add("Layer");
//Create graphics for layer
PdfGraphics graphics = layer.CreateGraphics(page);
//Draw ellipse
graphics.DrawEllipse(PdfPens.Red, new RectangleF(50, 50, 40, 40));
//Create square annotation
PdfSquareAnnotation annotation = new PdfSquareAnnotation(new RectangleF(200,
260, 50, 50), "Square annotation");
annotation.Color = new PdfColor(Color.FromArgb(0, 255, 0, 0));
//Set layer to annotation
annotation.Layer = layer;
//Add annotation to the created page
page.Annotations.Add(annotation);
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(memoryStream, "Sample.pdf");

```

ASP.NET CORE

```

//Create new PDF document
PdfDocument document = new PdfDocument();
//Add page
PdfPage page = document.Pages.Add();
//Add the layer
PdfLayer layer = document.Layers.Add("Layer");
//Create graphics for layer
PdfGraphics graphics = layer.CreateGraphics(page);
//Draw ellipse
graphics.DrawEllipse(PdfPens.Red, new RectangleF(50, 50, 40, 40));
//Create square annotation
PdfSquareAnnotation annotation = new PdfSquareAnnotation(new RectangleF(200,
260, 50, 50), "Square annotation");
annotation.Color = new PdfColor(Syncfusion.Drawing.Color.Red);
//Set layer to annotation
annotation.Layer = layer;
//Add annotation to the created page
page.Annotations.Add(annotation);
//Save and close the document
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);

```

```
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create new PDF document
PdfDocument document = new PdfDocument();
//Add page
PdfPage page = document.Pages.Add();
//Add the layer
PdfLayer layer = document.Layers.Add("Layer");
//Create graphics for layer
PdfGraphics graphics = layer.CreateGraphics(page);
//Draw ellipse
graphics.DrawEllipse(PdfPens.Red, new RectangleF(50, 50, 40, 40));
//Create square annotation
PdfSquareAnnotation annotation = new PdfSquareAnnotation(new RectangleF(200,
260, 50, 50), "Square annotation");
annotation.Color = new PdfColor(Syncfusion.Drawing.Color.Red);
//Set layer to annotation
annotation.Layer = layer;
//Add annotation to the created page
page.Annotations.Add(annotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pdf",
"application/pdf", stream);
}
```

The following code illustrates how to add annotation to the layers in an existing PDF document.

C#

```
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Gets the first page from the document
```



```

PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Add the layer
PdfLayer Layer = loadedDocument.Layers.Add("Layer");
//Create graphics for layer
PdfGraphics graphics = Layer.CreateGraphics(loadedPage);
//Draw ellipse
graphics.DrawEllipse(PdfPens.Red, new RectangleF(50, 50, 40, 40));
//Create square annotation
PdfSquareAnnotation annotation = new PdfSquareAnnotation(new RectangleF(200,
260, 50, 50), "Square annotation");
annotation.Color = new PdfColor(Color.Red);
//Set layer to annotation
annotation.Layer = Layer;
//Add annotation to the created page
loadedPage.Annotations.Add(annotation);
//Save the document
loadedDocument.Save("Output.pdf");
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the existing PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Gets the first page from the document
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Add the layer
Dim Layer As PdfLayer = loadedDocument.Layers.Add("Layer")
'Create graphics for layer
Dim graphics As PdfGraphics = Layer.CreateGraphics(loadedPage)
'Draw ellipse
graphics.DrawEllipse(PdfPens.Red, New RectangleF(50, 50, 40, 40))
'Create square annotation
Dim annotation As New PdfSquareAnnotation(New RectangleF(200, 260, 50, 50),
"Square annotation")
annotation.Color = New PdfColor(Color.Red)
'Set layer to annotation
annotation.Layer = Layer
'Add annotation to the created page
loadedPage.Annotations.Add(annotation)
'Save the document
loadedDocument.Save("Output.pdf")
'Close the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();

```

```

//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Gets the first page from the document
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Add the layer
PdfLayer Layer = loadedDocument.Layers.Add("Layer");
//Create graphics for layer
PdfGraphics graphics = Layer.CreateGraphics(loadedPage);
//Draw ellipse
graphics.DrawEllipse(PdfPens.Red, new RectangleF(50, 50, 40, 40));
//Create square annotation
PdfSquareAnnotation annotation = new PdfSquareAnnotation(new RectangleF(200, 260, 50, 50), "Square annotation");
annotation.Color = new PdfColor(Color.FromArgb(0, 255, 0, 0));
//Set layer to annotation
annotation.Layer = Layer;
//Add annotation to the created page
loadedPage.Annotations.Add(annotation);
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await loadedDocument.SaveAsync(memoryStream);
//Close the documents.
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples.
Save(memoryStream, "Sample.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open, FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Gets the first page from the document
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Add the layer
PdfLayer Layer = loadedDocument.Layers.Add("Layer");
//Create graphics for layer
PdfGraphics graphics = Layer.CreateGraphics(loadedPage);
//Draw ellipse
graphics.DrawEllipse(PdfPens.Red, new RectangleF(50, 50, 40, 40));
//Create square annotation
PdfSquareAnnotation annotation = new PdfSquareAnnotation(new RectangleF(200, 260, 50, 50), "Square annotation");
annotation.Color = new PdfColor(Syncfusion.Drawing.Color.Red);
//Set layer to annotation
annotation.Layer = Layer;
//Add annotation to the created page
loadedPage.Annotations.Add(annotation);
//Save and close the document
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document.
loadedDocument.Close(true);

```

```
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Gets the first page from the document
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Add the layer
PdfLayer Layer = loadedDocument.Layers.Add("Layer");
//Create graphics for layer
PdfGraphics graphics = Layer.CreateGraphics(loadedPage);
//Draw ellipse
graphics.DrawEllipse(PdfPens.Red, new RectangleF(50, 50, 40, 40));
//Create square annotation
PdfSquareAnnotation annotation = new PdfSquareAnnotation(new RectangleF(200,
260, 50, 50), "Square annotation");
annotation.Color = new PdfColor(Syncfusion.Drawing.Color.Red);
//Set layer to annotation
annotation.Layer = Layer;
//Add annotation to the created page
loadedPage.Annotations.Add(annotation);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pdf",
"application/pdf", stream);
}
```

Nested Layers

Essential PDF allows users to add nested layers in the PDF document. Refer to the following code snippet.

C#

```
//Create the PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
//Add the parent layer
PdfLayer layer = document.Layers.Add("Layer1");
PdfGraphics graphics = layer.CreateGraphics(page);
graphics.TranslateTransform(100, 60);
//Draw an arc
PdfPen pen = new PdfPen(Color.Red, 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add the child layer
PdfLayer layer2 = layer.Layers.Add("Layer2");
graphics = layer2.CreateGraphics(page);
graphics.TranslateTransform(100, 180);
//Draw an ellipse
graphics.DrawEllipse(pen, bounds);
//Save the document
document.Save("Output.pdf");
//Close the document
document.Close(true);
```

VB.NET

```
'Create the PDF document
Dim document As New PdfDocument()
Dim page As PdfPage = document.Pages.Add()
'Add the parent layer
Dim layer As PdfLayer = document.Layers.Add("Layer1")
Dim graphics As PdfGraphics = layer.CreateGraphics(page)
graphics.TranslateTransform(100, 60)
'Draw an arc
Dim pen As New PdfPen(Color.Red, 50)
Dim bounds As New RectangleF(0, 0, 50, 50)
graphics.DrawArc(pen, bounds, 360, 360)
'Add the child layer
Dim layer2 As PdfLayer = layer.Layers.Add("Layer2")
graphics = layer2.CreateGraphics(page)
graphics.TranslateTransform(100, 180)
'Draw an ellipse
graphics.DrawEllipse(pen, bounds)
'Save the document
document.Save("Output.pdf")
'Close the document
document.Close(True)
```

UWP

```
//Create the PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
//Add the parent layer
PdfLayer layer = document.Layers.Add("Layer1");
PdfGraphics graphics = layer.CreateGraphics(page);
```

```

graphics.TranslateTransform(100, 60);
//Draw an arc
PdfPen pen = new PdfPen(new PdfColor(255,0,0), 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add the child layer
PdfLayer layer2 = layer.Layers.Add("Layer2");
graphics = layer2.CreateGraphics(page);
graphics.TranslateTransform(100, 180);
//Draw an ellipse
graphics.DrawEllipse(pen, bounds);
//Save the document as stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document instances
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create the PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
//Add the parent layer
PdfLayer layer = document.Layers.Add("Layer1");
PdfGraphics graphics = layer.CreateGraphics(page);
graphics.TranslateTransform(100, 60);
//Draw an arc
PdfPen pen = new PdfPen(Color.Red, 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add the child layer
PdfLayer layer2 = layer.Layers.Add("Layer2");
graphics = layer2.CreateGraphics(page);
graphics.TranslateTransform(100, 180);
//Draw an ellipse
graphics.DrawEllipse(pen, bounds);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
document.Save(stream);
//If the position is not set to '0', the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create the PDF document
PdfDocument document = new PdfDocument();
PdfPage page = document.Pages.Add();
//Add the parent layer
PdfLayer layer = document.Layers.Add("Layer1");
PdfGraphics graphics = layer.CreateGraphics(page);
graphics.TranslateTransform(100, 60);
//Draw an arc
PdfPen pen = new PdfPen(Syncfusion.Drawing.Color.Red, 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add the child layer
PdfLayer layer2 = layer.Layers.Add("Layer2");
graphics = layer2.CreateGraphics(page);
graphics.TranslateTransform(100, 180);
//Draw an ellipse
graphics.DrawEllipse(pen, bounds);
//Save the document as stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document instances
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Removing layers from an existing PDF document

You can remove the layers from layer collection, represented by the [PdfPageLayerCollection](#) of the loaded page. This is illustrated in the following code sample.

C#

```

//Load the existing PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//Gets the first page from the document
PdfLoadedPage loadedPage = document.Pages[0] as PdfLoadedPage;
//Get the layer collection
PdfPageLayerCollection layers = loadedPage.Layers;
//Remove the layer
layers.RemoveAt(0);
//Save the document
document.Save("Output.pdf");

```

```
//Close the document
document.Close(true);
```

VB.NET

```
'Load the existing PDF document
Dim document As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Gets the first page from the document
Dim loadedPage As PdfLoadedPage = TryCast(document.Pages(0), PdfLoadedPage)
'Get the layer collection
Dim layers As PdfPageLayerCollection = loadedPage.Layers
'Remove the layer.
layers.RemoveAt(0)
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await document.OpenAsync(file);
//Gets the first page from the document
PdfLoadedPage loadedPage = document.Pages[0] as PdfLoadedPage;
//Get the layer collection
PdfPageLayerCollection layers = loadedPage.Layers;
//Remove the layer
layers.RemoveAt(0);
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(memoryStream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Gets the first page from the document
PdfLoadedPage loadedPage = document.Pages[0] as PdfLoadedPage;
//Get the layer collection
PdfPageLayerCollection layers = loadedPage.Layers;
```

```
//Remove the layer
layers.RemoveAt(0);
//Save and close the document
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Gets the first page from the document
PdfLoadedPage loadedPage = document.Pages[0] as PdfLoadedPage;
//Get the layer collection
PdfPageLayerCollection layers = loadedPage.Layers;
//Remove the layer
document.Layers.RemoveAt(0);
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}
```

Flattening the layers in an existing PDF document

You can flatten a layer in a PDF document by removing it from the [PdfDocumentLayerCollection](#). The following code snippet explains this.

C#


```
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Layers.pdf");
//Get the layer collection
PdfDocumentLayerCollection layers = loadedDocument.Layers;
//Flatten a layer in the PDF document
layers.RemoveAt(0);
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document
Dim loadedDocument As PdfLoadedDocument = New
PdfLoadedDocument("Layers.pdf")
'Get the layer collection
Dim layers As PdfDocumentLayerCollection = loadedDocument.Layers
'Flatten a layer in the PDF document
layers.RemoveAt(0)
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the instance of PdfLoadedDocument
loadedDocument.Close(True)
```

UWP

```
//Load the existing PDF document
Stream inputStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Layers.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the layer collection
PdfDocumentLayerCollection layers = loadedDocument.Layers;
//Flatten a layer in the PDF document
layers.RemoveAt(0);
//Create memory stream
MemoryStream stream = new MemoryStream();
//Open the document in browser after saving it
loadedDocument.Save(stream);
//Close the instance of PdfLoadedDocument
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the existing PDF document
FileStream inputStream = new FileStream("Layers.pdf", FileMode.Open);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the layer collection
PdfDocumentLayerCollection layers = loadedDocument.Layers;
//Flatten a layer in the PDF document
layers.RemoveAt(0);
```

```
//Saving the PDF to the MemoryStream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF document in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownloadName = "Output.pdf";
return fileStreamResult;
```

XAMARIN

```
//Load the existing PDF document
Stream inputStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Layers.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(inputStream);
//Get the layer collection
PdfDocumentLayerCollection layers = loadedDocument.Layers;
//Flatten a layer in the PDF document
layers.RemoveAt(0);
//Save the document to the stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Toggle the visibility of layers

The visibility of a layer can be mentioned while creating a new page layer.

The below code illustrates how to toggle the visibility of layers in new PDF document.

C#

```
//Create the document
PdfDocument document = new PdfDocument();
//Create a page
PdfPage page = document.Pages.Add();
//Add the first layer and enable the visibility.
PdfPageLayer layer = page.Layers.Add("Layer1", true);
```

```

PdfGraphics graphics = layer.Graphics;
graphics.TranslateTransform(100, 60);
//Draw Arc.
PdfPen pen = new PdfPen(System.Drawing.Color.Red, 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add another layer on the page and disable the visibility.
PdfPageLayer layer2 = page.Layers.Add("Layer2", false);
graphics = layer2.Graphics;
graphics.TranslateTransform(100, 180);
//Draw ellipse.
graphics.DrawEllipse(pen, bounds);
//Save the document.
document.Save("Sample.pdf");
//close the document
document.Close(true);

```

VB.NET

```

'Create the document
Dim document As New PdfDocument()
'Create a page
Dim page As PdfPage = document.Pages.Add()
'Add the first layer and enable the visibility.
Dim layer As PdfPageLayer = page.Layers.Add("Layer1", True)
Dim graphics As PdfGraphics = layer.Graphics
graphics.TranslateTransform(100, 60)
'Draw Arc.
Dim pen As New PdfPen(System.Drawing.Color.Red, 50)
Dim bounds As New RectangleF(0, 0, 50, 50)
graphics.DrawArc(pen, bounds, 360, 360)
'Add another layer on the page and disable the visibility.
Dim layer2 As PdfPageLayer = page.Layers.Add("Layer2", False)
graphics = layer2.Graphics
graphics.TranslateTransform(100, 180)
'Draw ellipse.
graphics.DrawEllipse(pen, bounds)
'Save the document.
document.Save("Sample.pdf")
'close the document
document.Close(True)

```

UWP

```

//Create the document
PdfDocument document = new PdfDocument();
//Create a page
PdfPage page = document.Pages.Add();
//Add the first layer and enable the visibility.
PdfPageLayer layer = page.Layers.Add("Layer1", true);
PdfGraphics graphics = layer.Graphics;
graphics.TranslateTransform(100, 60);
//Draw Arc.
PdfPen pen = new PdfPen(new PdfColor(255, 0, 0), 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);

```

```
//Add another layer on the page and disable the visibility.
PdfPageLayer layer2 = page.Layers.Add("Layer2", false);
graphics = layer2.Graphics;
graphics.TranslateTransform(100, 180);
//Draw ellipse.
graphics.DrawEllipse(pen, bounds);
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(memoryStream, "Output.pdf");
```

ASP.NET CORE

```
//Create the document
PdfDocument document = new PdfDocument();
//Create a page
PdfPage page = document.Pages.Add();
//Add the first layer and enable the visibility.
PdfPageLayer layer = page.Layers.Add("Layer1", true);
PdfGraphics graphics = layer.Graphics;
graphics.TranslateTransform(100, 60);
//Draw Arc.
PdfPen pen = new PdfPen(Syncfusion.Drawing.Color.Red, 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add another layer on the page and disable the visibility.
PdfPageLayer layer2 = page.Layers.Add("Layer2", false);
graphics = layer2.Graphics;
graphics.TranslateTransform(100, 180);
//Draw ellipse.
graphics.DrawEllipse(pen, bounds);
//Save and close the document
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create the document
PdfDocument document = new PdfDocument();
//Create a page
PdfPage page = document.Pages.Add();
//Add the first layer and enable the visibility.
```

```

PdfPageLayer layer = page.Layers.Add("Layer1", true);
PdfGraphics graphics = layer.Graphics;
graphics.TranslateTransform(100, 60);
//Draw Arc.
PdfPen pen = new PdfPen(Syncfusion.Drawing.Color.Red, 50);
RectangleF bounds = new RectangleF(0, 0, 50, 50);
graphics.DrawArc(pen, bounds, 360, 360);
//Add another layer on the page and disable the visibility.
PdfPageLayer layer2 = page.Layers.Add("Layer2", false);
graphics = layer2.Graphics;
graphics.TranslateTransform(100, 180);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pdf",
"application/pdf", stream);
}

```

The following code illustrates how to toggle the visibility of layers in an existing PDF document by disabling the [Visible](#) property of [PdfLayer](#).

C#

```

//Load the existing PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Input.pdf");
//Gets the first layer from the layer collection
PdfLayer layer = document.Layers[0];
//Disable the visibility
layer.Visible = false;
//Save the document
document.Save("Output.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Load the existing PDF document
Dim document As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Get the first layer from the layer collection
Dim layer As PdfLayer = document.Layers(0)
'Disable the visibility
layer.Visible = False

```

```
'Save the document
document.Save("Output.pdf")
'Close the document
document.Close(True)
```

UWP

```
//Load the PDF document as stream
Stream pdfStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Input.pdf");
//Creates an empty PDF loaded document instance
PdfLoadedDocument document = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of
PdfLoadedDocument class
await document.OpenAsync(pdfStream);
//Gets the first layer from the layer collection
PdfLayer layer = document.Layers[0];
//Disable the visibility
layer.Visible = false;
MemoryStream memoryStream = new MemoryStream();
//Save the document.
await document.SaveAsync(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples.
Save(memoryStream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Gets the first layer from the layer collection
PdfLayer layer = document.Layers[0];
//Disable the visibility
layer.Visible = false;
//Save and close the document
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document.
document.Close(true);
//Defining the ContentType for pdf file.
string contentType = "application/pdf";
//Define the file name.
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name.
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument document = new PdfLoadedDocument(docStream);
//Gets the first layer from the layer collection
PdfLayer layer = document.Layers[0];
//Disable the visibility
layer.Visible = false;
//Save the document into stream.
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}
```

Tagged PDF

Introduction

The Tagged PDF is a PDF that includes structure in terms for a set of instruction that defines reading order and meaning of significant elements such as figures, images, lists, tables and more. Tagged PDF documents created using Syncfusion PDF library are compliant with section 508 (PDF/UA) standard or WCAG 2.0 standard (ISO 14289-1:2014).

Usually tagged PDF used to making content accessible to users who rely on assistive technology.

This section explains how to add tags to PDF elements such as text element, image, shapes, form fields, annotations, table, list, and more

Adding tag to text element

You can add tag to text or paragraphs in PDF document by using the [PdfTag](#) property available in the [PdfTextElement](#) class and specifying the tag type as [Paragraph](#) of [PdfTagType](#) Enum in the [PdfStructureElement](#) class.

The following code sample explains you how to add tag for the text element in PDF document.

C#

```
//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "PdfTextElement";
```

```

//Creates new page
PdfPage page = doc.Pages.Add();
//Initialize the structure element with tag type paragraph
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Paragraph);
//represents the text that is exact replacement for PdfTextElement
structureElement.ActualText = "Simple paragraph element";
string text = "Adventure Works Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Initialize the PDF text element
PdfTextElement element = new PdfTextElement(text);
//Adding tag to the text element
element.PdfTag = structureElement;
//Creates font for the text element
element.Font = new PdfStandardFont(PdfFontFamily.TimesRoman, 12);
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Draws text
PdfLayoutResult result = element.Draw(page, new RectangleF(0, 0,
page.Graphics.ClientSize.Width, 200));
//Save the document and dispose it
doc.Save("Output.pdf");
doc.Close(true);

```

VB.NET

```

'Creates new PDF document
Dim doc As PdfDocument = New PdfDocument()
'Set the document title
doc.DocumentInformation.Title = "PdfTextElement"
'Creates new page
Dim page As PdfPage = doc.Pages.Add()
'Initialize the structure element with tag type paragraph
Dim structureElement As PdfStructureElement = New
PdfStructureElement(PdfTagType.Paragraph)
'represents the text that is exact replacement for PdfTextElement
structureElement.ActualText = "Simple paragraph element"
Dim text As String = "Adventure Works Cycles, the fictitious company on
which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company. The company manufactures and sells
metal and composite bicycles to North American, European and Asian
commercial markets. While its base operation is located in Washington with
290 employees, several regional sales teams are located throughout their
market base."
'Initialize the PDF text element
Dim element As PdfTextElement = New PdfTextElement(text)
'Adding tag to the text element
element.PdfTag = structureElement
'Creates font for the text element
element.Font = New PdfStandardFont(PdfFontFamily.TimesRoman, 12)
element.Brush = New PdfSolidBrush(New PdfColor(89, 89, 93))
'Draws text

```



```

Dim result As PdfLayoutResult = element.Draw(page, New RectangleF(0, 0,
page.Graphics.ClientSize.Width, 200))
'Save the document and dispose it
doc.Save("Output.pdf")
doc.Close(True)

```

UWP

```

//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "PdfTextElement";
//Creates new page
PdfPage page = doc.Pages.Add();
//Initialize the structure element with tag type paragraph.
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Paragraph);
//Represents the text that is exact replacement for PdfTextElement
structureElement.ActualText = "Simple paragraph element";
string text = "Adventure Works Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Initialize the PDF text element
PdfTextElement element = new PdfTextElement(text);
//Adding tag to the text element
element.PdfTag = structureElement;
//Creates font for the text element
element.Font = new PdfStandardFont(PdfFontFamily.TimesRoman, 12);
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Draws text
PdfLayoutResult result = element.Draw(page, new RectangleF(0, 0,
page.Graphics.ClientSize.Width, 200));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "PdfTextElement";
//Creates new page
PdfPage page = doc.Pages.Add();
//Initialize the structure element with tag type paragraph
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Paragraph);
//Represents the text that is exact replacement for PdfTextElement

```

```

structureElement.ActualText = "Simple paragraph element";
string text = "Adventure Works Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Initialize the PDF text element
PdfTextElement element = new PdfTextElement(text);
//Adding tag to the text element
element.PdfTag = structureElement;
//Creates font for the text element
element.Font = new PdfStandardFont(PdfFontFamily.TimesRoman, 12);
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Draws text
PdfLayoutResult result = element.Draw(page, new RectangleF(0, 0,
page.Graphics.ClientSize.Width, 200));
//Save the document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
stream.Position = 0;
//Closes the document
doc.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "PdfTextElement";
//Creates new page
PdfPage page = doc.Pages.Add();
//Initialize the structure element with tag type paragraph
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Paragraph);
//represents the text that is exact replacement for PdfTextElement
structureElement.ActualText = "Simple paragraph element";
string text = "Adventure Works Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Initialize the PDF text element
PdfTextElement element = new PdfTextElement(text);
//Adding tag to the text element.
element.PdfTag = structureElement;
//Creates font for the text element
element.Font = new PdfStandardFont(PdfFontFamily.TimesRoman, 12);

```

```

element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Draws text
PdfLayoutResult result = element.Draw(page, new RectangleF(0, 0,
page.Graphics.ClientSize.Width, 200));
//Save the document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Adding tag to image

You can add tag to image in the PDF document by using the [PdfTag](#) property available in the [PdfBitmap](#) class and specifying the tag type as [Figure](#) of [PdfTagType](#) Enum in the [PdfStructureElement](#) class. You can add alternate text to image by using the [AlternateText](#) property available in the [PdfStructureElement](#) class.

The following code explains how to add tag for image element in PDF document.

C#

```

//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "Image";
//Creates new page
PdfPage page = doc.Pages.Add();
//Draw string
page.Graphics.DrawString("JPEG Image:", new
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, new PointF(0, 0));
//Create a new PDF bitmap object
PdfBitmap bitmap = new PdfBitmap("syncfusion.jpg");
//Set the tag type
PdfStructureElement imageElement = new
PdfStructureElement(PdfTagType.Figure);
//Set the alternate text
imageElement.AlternateText = "GreenTree";
//adding tag to the PDF image
bitmap.PdfTag = imageElement;
//Draw image
bitmap.Draw(page.Graphics, new PointF(50, 20));

```

```
//Save the document and dispose it
doc.Save("Image.pdf");
doc.Close(true);
```

VB.NET

```
'Creates new PDF document
Dim doc As PdfDocument = New PdfDocument()
'Set the document title
doc.DocumentInformation.Title = "Image"
'Creates new page
Dim page As PdfPage = doc.Pages.Add()
'Draw string
page.Graphics.DrawString("JPEG Image:", New
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, New PointF(0, 0))
'Create a new PDF bitmap object
Dim bitmap As PdfBitmap = New PdfBitmap("syncfusion.jpg")
'Set the tag type
Dim imageElement As PdfStructureElement = New
PdfStructureElement(PdfTagType.Figure)
'Set the alternate text
imageElement.AlternateText = "GreenTree"
'adding tag to the PDF image
bitmap.PdfTag = imageElement
'Draw image
bitmap.Draw(page.Graphics, New PointF(50, 20))
'Save the document and dispose it
doc.Save("Image.pdf")
doc.Close(True)
```

UWP

```
//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "Image";
//Creates new page
PdfPage page = doc.Pages.Add();
//Draw string
page.Graphics.DrawString("JPEG Image:", new
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, new PointF(0, 0));
//Load the image as stream
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.syncfusion.jpg");
//Create a new PDF bitmap object
PdfBitmap bitmap = new PdfBitmap(imageStream);
//Set the tag type
PdfStructureElement imageElement = new
PdfStructureElement(PdfTagType.Figure);
//Set the alternate text
imageElement.AlternateText = "GreenTree";
//Adding tag to the PDF image
bitmap.PdfTag = imageElement;
```

```
//Draw image
bitmap.Draw(page.Graphics, new PointF(50, 20));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Image.pdf");
```

ASP.NET CORE

```
//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "Image";
//Creates new page
PdfPage page = doc.Pages.Add();
//Draw string
page.Graphics.DrawString("JPEG Image:", new
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, new PointF(0, 0));
//Load the image as stream
FileStream imageStream = new FileStream("syncfusion.jpg", FileMode.Open,
FileAccess.Read);
//Create a new PDF bitmap object
PdfBitmap bitmap = new PdfBitmap(imageStream);
//Set the tag type
PdfStructureElement imageElement = new
PdfStructureElement(PdfTagType.Figure);
//Set the alternate text
imageElement.AlternateText = "GreenTree";
//adding tag to the PDF image
bitmap.PdfTag = imageElement;
//Draw image
bitmap.Draw(page.Graphics, new PointF(50, 20));
//Save the document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
stream.Position = 0;
//Closes the document
doc.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Image.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
```

```

doc.DocumentInformation.Title = "Image";
//Creates new page
PdfPage page = doc.Pages.Add();
//Draw string
page.Graphics.DrawString("JPEG Image:", new
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, new PointF(0, 0));
//Load the file as stream
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
syncfusion.jpg");
//Create a new PDF bitmap object
PdfBitmap bitmap = new PdfBitmap(imageStream);
//Set the tag type
PdfStructureElement imageElement = new
PdfStructureElement(PdfTagType.Figure);
//Set the alternate text
imageElement.AlternateText = "GreenTree";
//adding tag to the PDF image
bitmap.PdfTag = imageElement;
//Draw image
bitmap.Draw(page.Graphics, new PointF(50, 20));
//Save the document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Image.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Image.pdf",
"application/pdf", stream);
}

```

Adding tag to shapes

You can add tag to shapes such as rectangle, line, circle, polygon, and more by using the [PdfTag](#) property and specifying the tag type as [Figure](#) of [PdfTagType](#) Enum. You can add alternate text to shapes by using the [AlternateText](#) property available in the [PdfStructureElement](#) class.

The following code explains how to add tag for shape element in the PDF document.

C#

```

//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "LineShape";

```

```

//Add new page
PdfPage page = doc.Pages.Add();
//Draw text
page.Graphics.DrawString("Line Shape:", new
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, new PointF(30, 80));
//Initialize structure element with tag type as Figure
PdfStructureElement element = new PdfStructureElement(PdfTagType.Figure);
//Set alternate text
element.AlternateText = "Line Sample";
//Initialize the line shape
PdfLine line = new PdfLine(100, 100, 100, 300);
line.Pen = new PdfPen(Color.Red);
//Adding tag to the line element
line.PdfTag = element;
//Draws the line
line.Draw(page.Graphics);
//Save the document and dispose it
doc.Save("Output.pdf");

```

VB.NET

```

'Creates new PDF document
Dim doc As PdfDocument = New PdfDocument()
'Set the document title
doc.DocumentInformation.Title = "LineShape"
'Add new page
Dim page As PdfPage = doc.Pages.Add()
'Draw text
page.Graphics.DrawString("Line Shape:", New
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, New PointF(30, 80))
'Initialize structure element with tag type as Figure
Dim element As PdfStructureElement = New
PdfStructureElement(PdfTagType.Figure)
'Set alternate text
element.AlternateText = "Line Sample"
'Initialize the line shape
Dim line As PdfLine = New PdfLine(100, 100, 100, 300)
line.Pen = New PdfPen(Color.Red)
'Adding tag to the line element
line.PdfTag = element
'Draws the line
line.Draw(page.Graphics)
'Save the document and dispose it
doc.Save("Output.pdf")

```

UWP

```

//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "LineShape";
//Add new page
PdfPage page = doc.Pages.Add();
//Draw text

```

```

page.Graphics.DrawString("Line Shape:", new
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, new PointF(30, 80));
//Initialize structure element with tag type as Figure
PdfStructureElement element = new PdfStructureElement(PdfTagType.Figure);
//Set alternate text
element.AlternateText = "Line Sample";
//Initialize the line shape
PdfLine line = new PdfLine(100, 100, 100, 300);
line.Pen = new PdfPen(new PdfColor(255, 0, 0));
//Adding tag to the line element
line.PdfTag = element;
//Draws the line
line.Draw(page.Graphics);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "LineShape";
//Add new page
PdfPage page = doc.Pages.Add();
//Draw text
page.Graphics.DrawString("Line Shape:", new
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, new PointF(30, 80));
//Initialize structure element with tag type as figure
PdfStructureElement element = new PdfStructureElement(PdfTagType.Figure);
//Set alternate text
element.AlternateText = "Line Sample";
//Initialize the line shape
PdfLine line = new PdfLine(100, 100, 100, 300);
line.Pen = new PdfPen(Color.Red);
//Adding tag to the line element
line.PdfTag = element;
//Draws the line
line.Draw(page.Graphics);
//Save the document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
stream.Position = 0;
//Closes the document
doc.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";

```



```
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "LineShape";
//Add new page
PdfPage page = doc.Pages.Add();
//Draw text
page.Graphics.DrawString("Line Shape:", new
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, new PointF(30, 80));
//Initialize structure element with tag type as figure
PdfStructureElement element = new PdfStructureElement(PdfTagType.Figure);
//Set alternate text
element.AlternateText = "Line Sample";
//Initialize the line shape
PdfLine line = new PdfLine(100, 100, 100, 300);
line.Pen = new PdfPen(Syncfusion.Drawing.Color.Red);
//Adding tag to the line element
line.PdfTag = element;
//Draws the line
line.Draw(page.Graphics);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Adding tag to Form Fields

You can tag the form fields in the PDF document by using the [PdfTag](#) and specifying the tag type as Form of [PdfTagType](#) Enum.

The following code explains how to add tag for the form fields in PDF document.

C#

```

//Creates new PDF document
PdfDocument doc = new PdfDocument();
doc.DocumentInformation.Title = "Form Fields";
//Adds new page
PdfPage page = doc.Pages.Add();
// Create a text box field
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "This is form field
text box");
//Adding tag to the text box field
textBoxField.PdfTag = new PdfStructureElement(PdfTagType.Form);
textBoxField.Text = "Filled text box";
//Set properties to the text box
textBoxField.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 12);
textBoxField.BorderColor = new PdfColor(Color.Gray);
textBoxField.BorderStyle = PdfBorderStyle.Beveled;
textBoxField.Bounds = new RectangleF(200, 0, 90, 20);
textBoxField.ToolTip = "TextBox field";
doc.Form.Fields.Add(textBoxField);
//Save the document and dispose it
doc.Save("Output.pdf");
doc.Close(true);

```

VB.NET

```

'Creates new PDF document
Dim doc As PdfDocument = New PdfDocument()
doc.DocumentInformation.Title = "Form Fields"
'Adds new page
Dim page As PdfPage = doc.Pages.Add()
' Create a text box field
Dim textBoxField As PdfTextBoxField = New PdfTextBoxField(page, "This is
form field text box")
'Adding tag to the text box field
textBoxField.PdfTag = New PdfStructureElement(PdfTagType.Form)
textBoxField.Text = "Filled text box"
'Set properties to the text box
textBoxField.Font = New PdfStandardFont(PdfFontFamily.Helvetica, 12)
textBoxField.BorderColor = New PdfColor(Color.Gray)
textBoxField.BorderStyle = PdfBorderStyle.Beveled
textBoxField.Bounds = New RectangleF(200, 0, 90, 20)
textBoxField.ToolTip = "TextBox field"
doc.Form.Fields.Add(textBoxField)
'Save the document and dispose it
doc.Save("Output.pdf")
doc.Close(True)

```

UWP

```

//Creates new PDF document
PdfDocument doc = new PdfDocument();
doc.DocumentInformation.Title = "Form Fields";
//Adds new page
PdfPage page = doc.Pages.Add();
// Create a text box field
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "This is form field
text box");

```

```

//Adding tag to the text box field
textBoxField.PdfTag = new PdfStructureElement(PdfTagType.Form);
textBoxField.Text = "Filled text box";
//Set properties to the text box
textBoxField.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 12);
textBoxField.BorderColor = new PdfColor(128,128,128);
textBoxField.BorderStyle = PdfBorderStyle.Beveled;
textBoxField.Bounds = new RectangleF(200, 0, 90, 20);
textBoxField.ToolTip = "TextBox field";
doc.Form.Fields.Add(textBoxField);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Creates new PDF document
PdfDocument doc = new PdfDocument();
doc.DocumentInformation.Title = "Form Fields";
//Adds new page
PdfPage page = doc.Pages.Add();
// Create a text box field
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "This is form field
text box");
//Adding tag to the text box field
textBoxField.PdfTag = new PdfStructureElement(PdfTagType.Form);
textBoxField.Text = "Filled text box";
//Set properties to the text box
textBoxField.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 12);
textBoxField.BorderColor = new PdfColor(Color.Gray);
textBoxField.BorderStyle = PdfBorderStyle.Beveled;
textBoxField.Bounds = new RectangleF(200, 0, 90, 20);
textBoxField.ToolTip = "TextBox field";
doc.Form.Fields.Add(textBoxField);
//Save the document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
stream.Position = 0;
//Closes the document
doc.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates new PDF document

```

```

PdfDocument doc = new PdfDocument();
doc.DocumentInformation.Title = "Form Fields";
//Adds new page
PdfPage page = doc.Pages.Add();
// Create a text box field
PdfTextBoxField textBoxField = new PdfTextBoxField(page, "This is form field
text box");
//Adding tag to the text box field
textBoxField.PdfTag = new PdfStructureElement(PdfTagType.Form);
textBoxField.Text = "Filled text box";
//Set properties to the text box
textBoxField.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 12);
textBoxField.BorderColor = new PdfColor(Syncfusion.Drawing.Color.Gray);
textBoxField.BorderStyle = PdfBorderStyle.Beveled;
textBoxField.Bounds = new RectangleF(200, 0, 90, 20);
textBoxField.ToolTip = "TextBox field";
doc.Form.Fields.Add(textBoxField);
//Save the document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Adding tag to Annotation

You can add tags to annotation in PDF document by using the [PdfTag](#) property and specifying the tag type as **Annotation** of [PdfTagType](#) Enum.

The following code explains how to add tag for the annotations in PDF document.

C#

```

//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "LineShape";
//Add new page
PdfPage page = doc.Pages.Add();
//Initialize the structure element with tag type as annotation
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Annotation);
structureElement.AlternateText = "Popup Annotation";

```

```

RectangleF rectangle = new RectangleF(10, 40, 30, 30);
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test
popup annotation");
//Adding tag for the annotation
popupAnnotation.PdfTag = structureElement;
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the PDF pop-up icon
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds this annotation to a new page
page.Annotations.Add(popupAnnotation);
//Saves the document to disk
doc.Save("PopupAnnotation.pdf");
//Close the PDF document
doc.Close(true);

```

VB.NET

```

'Creates new PDF document
Dim doc As PdfDocument = New PdfDocument()
'Set the document title
doc.DocumentInformation.Title = "LineShape"
'Add new page
Dim page As PdfPage = doc.Pages.Add()
'Initialize the structure element with tag type as annotation
Dim structureElement As PdfStructureElement = New
PdfStructureElement(PdfTagType.Annotation)
structureElement.AlternateText = "Popup Annotation"
Dim rectangle As RectangleF = New RectangleF(10, 40, 30, 30)
Dim popupAnnotation As PdfPopupAnnotation = New
PdfPopupAnnotation(rectangle, "Test popup annotation")
'Adding tag for the annotation
popupAnnotation.PdfTag = structureElement
popupAnnotation.Border.Width = 4
popupAnnotation.Border.HorizontalRadius = 20
popupAnnotation.Border.VerticalRadius = 30
'Sets the PDF pop-up icon
popupAnnotation.Icon = PdfPopupIcon.NewParagraph
'Adds this annotation to a new page
page.Annotations.Add(popupAnnotation)
'Saves the document to disk
doc.Save("PopupAnnotation.pdf")
'Close the PDF document
doc.Close(True)

```

UWP

```

//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "LineShape";
//Add new page
PdfPage page = doc.Pages.Add();
//Initialize the structure element with tag type as annotation

```

```

PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Annotation);
structureElement.AlternateText = "Popup Annotation";
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test
popup annotation");
//Adding tag for the annotation
popupAnnotation.PdfTag = structureElement;
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the PDF pop-up icon
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds this annotation to a new page
page.Annotations.Add(popupAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
//Close the document
doc.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "PopupAnnotation.pdf");

```

ASP.NET CORE

```

//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "LineShape";
//Add new page
PdfPage page = doc.Pages.Add();
//Initialize the structure element with tag type as annotation
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Annotation);
structureElement.AlternateText = "Popup Annotation";
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test
popup annotation"
//Adding tag for the annotation
popupAnnotation.PdfTag = structureElement;
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the PDF pop-up icon
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds this annotation to a new page
page.Annotations.Add(popupAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
stream.Position = 0;
//Closes the document
doc.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";

```

```
//Define the file name
string fileName = "PopupAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates new PDF document
PdfDocument doc = new PdfDocument();
//Set the document title
doc.DocumentInformation.Title = "LineShape";
//Add new page
PdfPage page = doc.Pages.Add();
//Initialize the structure element with tag type as annotation
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Annotation);
structureElement.AlternateText = "Popup Annotation";
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle, "Test
popup annotation");
//Adding tag for the annotation
popupAnnotation.PdfTag = structureElement;
popupAnnotation.Border.Width = 4;
popupAnnotation.Border.HorizontalRadius = 20;
popupAnnotation.Border.VerticalRadius = 30;
//Sets the PDF pop-up icon
popupAnnotation.Icon = PdfPopupIcon.NewParagraph;
//Adds this annotation to a new page
page.Annotations.Add(popupAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("PopupAnnotati
on.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("PopupAnnotation.pdf",
"application/pdf", stream);
}
```

Adding tag to Hyperlink

You can tag the hyperlink present in the PDF document by using [PdfTag](#) available in the [PdfTextWebLink](#) class and specifying the tag type as [Link](#) of [PdfTagType](#) Enum.

The following code example shows how to add tag for hyperlink in PDF document

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
document.DocumentInformation.Title = "Link";
//Add a page to the document
PdfPage page = document.Pages.Add();
//Creates new PDF structure element with tag type link
PdfStructureElement linkStructureElement = new
PdfStructureElement(PdfTagType.Link);
//Create the font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12f);
//Create the text web link
PdfTextWebLink textLink = new PdfTextWebLink();
//Adding tag to text web link
textLink.PdfTag = linkStructureElement;
//Set the hyperlink
textLink.Url = "http://www.syncfusion.com";
//Set the link text
textLink.Text = "Syncfusion .NET components and controls";
//Set the font
textLink.Font = font;
textLink.Brush = PdfBrushes.Blue;
//Draw the hyperlink in PDF page
textLink.DrawTextWebLink(page, new PointF(10, 40));
//Save the document
document.Save("Output.pdf");
//Close the document
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As PdfDocument = New PdfDocument()
document.DocumentInformation.Title = "Link"
'Add a page to the document
Dim page As PdfPage = document.Pages.Add()
'Creates new PDF structure element with tag type link
Dim linkStructureElement As PdfStructureElement = New
PdfStructureElement(PdfTagType.Link)
'Create the font
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 12.0F)
'Create the text web link
Dim textLink As PdfTextWebLink = New PdfTextWebLink()
'Adding tag to text web link
textLink.PdfTag = linkStructureElement
'Set the hyperlink
textLink.Url = "http://www.syncfusion.com"
'Set the link text
textLink.Text = "Syncfusion .NET components and controls"
'Set the font
textLink.Font = font
textLink.Brush = PdfBrushes.Blue
'Draw the hyperlink in PDF page
textLink.DrawTextWebLink(page, New PointF(10, 40))
```



```
'Save the document
document.Save("Output.pdf")
'Close the document
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
document.DocumentInformation.Title = "Link";
//Add a page to the document
PdfPage page = document.Pages.Add();
//Creates new PDF structure element with tag type link
PdfStructureElement linkStructureElement = new
PdfStructureElement(PdfTagType.Link);
//Create the font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12f);
//Create the text web link
PdfTextWebLink textLink = new PdfTextWebLink();
//Adding tag to text web link
textLink.PdfTag = linkStructureElement;
//Set the hyperlink
textLink.Url = "http://www.syncfusion.com";
//Set the link text
textLink.Text = "Syncfusion .NET components and controls";
//Set the font
textLink.Font = font;
textLink.Brush = PdfBrushes.Blue;
//Draw the hyperlink in PDF page
textLink.DrawTextWebLink(page, new PointF(10, 40));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
document.DocumentInformation.Title = "Link";
//Add a page to the document
PdfPage page = document.Pages.Add();
//Creates new PDF structure element with tag type link
PdfStructureElement linkStructureElement = new
PdfStructureElement(PdfTagType.Link);
//Create the font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12f);
//Create the text web link
PdfTextWebLink textLink = new PdfTextWebLink();
//Adding tag to text web link
textLink.PdfTag = linkStructureElement;
//Set the hyperlink
```

```

textLink.Url = "http://www.syncfusion.com";
//Set the link text
textLink.Text = "Syncfusion .NET components and controls";
//Set the font
textLink.Font = font;
textLink.Brush = PdfBrushes.Blue;
//Draw the hyperlink in PDF page
textLink.DrawTextWebLink(page, new PointF(10, 40));
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
document.DocumentInformation.Title = "Link";
//Add a page to the document
PdfPage page = document.Pages.Add();
//Creates new pdf structure element with tag type link
PdfStructureElement linkStructureElement = new
PdfStructureElement(PdfTagType.Link);
//Create the font
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 12f);
//Create the text web link
PdfTextWebLink textLink = new PdfTextWebLink();
//Adding tag to text web link
textLink.PdfTag = linkStructureElement;
//Set the hyperlink
textLink.Url = "http://www.syncfusion.com";
//Set the link text
textLink.Text = "Syncfusion .NET components and controls";
//Set the font
textLink.Font = font;
textLink.Brush = PdfBrushes.Blue;
//Draw the hyperlink in PDF page
textLink.DrawTextWebLink(page, new PointF(10, 40));
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Adding tag to Template

You can add tags to template in PDF document by using the [PdfTag](#) property available in the [PdfTemplate](#) class.

The following code sample explains how to add tag support for the template element.

C#

```

//Creates a new PDF document
PdfDocument pdfDocument = new PdfDocument();
pdfDocument.DocumentInformation.Title = "TemplateDocument";
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
pdfPage.Graphics.DrawString("Rectangle:", new
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, new PointF(0, 0));
//Create a PDF template
PdfTemplate template = new PdfTemplate(100, 50);
//Initialize the structure element with tag type figure
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Figure);
//Set alternative description for figure
structureElement.AlternateText = "Template Figure";
//Adding tag to the template element
template.PdfTag = structureElement;
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
PdfBrush brush = new PdfSolidBrush(Color.Pink);
//Draw rectangle using template graphics
template.Graphics.DrawRectangle(brush, new RectangleF(0, 30, 150, 90));
//Draw the template on the page graphics of the document
pdfPage.Graphics.DrawPdfTemplate(template, PointF.Empty);
//Save the document and dispose it
pdfDocument.Save("Output.pdf");
pdfDocument.Close(true);

```

VB.NET

```

'Creates a new PDF document
Dim pdfDocument As PdfDocument = New PdfDocument()
pdfDocument.DocumentInformation.Title = "TemplateDocument"
'Add a page to the PDF document
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()

```

```

pdfPage.Graphics.DrawString("Rectangle:", New
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, New PointF(0, 0))
'Create a PDF template
Dim template As PdfTemplate = New PdfTemplate(100, 50)
'Initialize the structure element with tag type figure
Dim structureElement As PdfStructureElement = New
PdfStructureElement(PdfTagType.Figure)
'Set alternative description for figure
structureElement.AlternateText = "Template Figure"
'Adding tag to the template element
template.PdfTag = structureElement
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 14)
Dim brush As PdfBrush = New PdfSolidBrush(Color.Pink)
'Draw rectangle using template graphics
template.Graphics.DrawRectangle(brush, New RectangleF(0, 30, 150, 90))
'Draw the template on the page graphics of the document
pdfPage.Graphics.DrawPdfTemplate(template, PointF.Empty)
'Save the document and dispose it
pdfDocument.Save("Output.pdf")
pdfDocument.Close(True)

```

UWP

ASP.NET CORE

```

//Creates a new PDF document
PdfDocument pdfDocument = new PdfDocument();
pdfDocument.DocumentInformation.Title = "TemplateDocument";
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
pdfPage.Graphics.DrawString("Rectangle:", new
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, new PointF(0, 0));
//Create a PDF template
PdfTemplate template = new PdfTemplate(100, 50);
//Initialize the structure element with tag type figure
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Figure);
//Set alternative description for figure
structureElement.AlternateText = "Template Figure";
//Adding tag to the template element
template.PdfTag = structureElement;
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
PdfBrush brush = new PdfSolidBrush(Color.Pink);
//Draw rectangle using template graphics
template.Graphics.DrawRectangle(brush, new RectangleF(0, 30, 150, 90));
//Draw the template on the page graphics of the document
pdfPage.Graphics.DrawPdfTemplate(template, PointF.Empty);
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;

```

```
//Closes the document
pdfDocument.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document
PdfDocument pdfDocument = new PdfDocument();
pdfDocument.DocumentInformation.Title = "TemplateDocument";
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
pdfPage.Graphics.DrawString("Rectangle:", new
PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold),
PdfBrushes.Blue, new PointF(0, 0));
//Create a PDF template
PdfTemplate template = new PdfTemplate(100, 50);
//Initialize the structure element with tag type figure
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Figure);
//Set alternative description for figure
structureElement.AlternateText = "Template Figure";
//Adding tag to the template element
template.PdfTag = structureElement;
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 14);
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Pink);
//Draw rectangle using template graphics
template.Graphics.DrawRectangle(brush, new RectangleF(0, 30, 150, 90));
//Draw the template on the page graphics of the document
pdfPage.Graphics.DrawPdfTemplate(template, PointF.Empty);
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Close the document
pdfDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Adding tag to Table

You can tag the table in the PDF document by specifying the tag type as **Table** of [PdfTagType](#) Enum. The following tag types are used to mention the table header, rows, and cells:

1. PdfTagType.TableHeader
2. PdfTagType.TableRow
3. PdfTagType.TableDataCell

The following code snippet illustrates how to add tag for table element.

C#

```
//Creates a new PDF document
PdfDocument pdfDocument = new PdfDocument();
pdfDocument.DocumentInformation.Title = "Table";
//Adds new page
PdfPage pdfPage = pdfDocument.Pages.Add();
//Initialize the new structure element with tag type table
PdfStructureElement element = new PdfStructureElement(PdfTagType.Table);
//Create a new PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Adding tag to PDF grid
pdfGrid.PdfTag = element;
//Add three columns
pdfGrid.Columns.Add(3);
//Add header
pdfGrid.Headers.Add(1);
PdfGridRow pdfGridHeader = pdfGrid.Headers[0];
pdfGridHeader.Style.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 12, PdfFontStyle.Bold);
pdfGridHeader.Style.TextBrush = PdfBrushes.Brown;
//Adding tag for each row with tag type TR
pdfGridHeader.PdfTag = new PdfStructureElement(PdfTagType.TableRow);
pdfGridHeader.Cells[0].Value = "Employee ID";
//Adding tag for header cell with tag type TH
pdfGridHeader.Cells[0].PdfTag = new PdfStructureElement(PdfTagType.TableHeader);
pdfGridHeader.Cells[1].Value = "Employee Name";
//Adding tag for header cell with tag type TH
pdfGridHeader.Cells[1].PdfTag = new PdfStructureElement(PdfTagType.TableHeader);
pdfGridHeader.Cells[2].Value = "Salary";
//Adding tag for header cell with tag type TH
pdfGridHeader.Cells[2].PdfTag = new PdfStructureElement(PdfTagType.TableHeader);
//Add rows.
PdfGridRow pdfGridRow = pdfGrid.Rows.Add();
pdfGridRow.PdfTag = new PdfStructureElement(PdfTagType.TableRow);
pdfGridRow.Cells[0].Value = "E01";
pdfGridRow.Cells[1].Value = "Clay";
pdfGridRow.Cells[2].Value = "$10,000";
//Adding tag for each cell with tag type TD
pdfGridRow.Cells[0].PdfTag = new PdfStructureElement(PdfTagType.TableDataCell);
```

```
pdfGridRow.Cells[1].PdfTag = new
PdfStructureElement(PdfTagType.TableDataCell);
pdfGridRow.Cells[2].PdfTag = new
PdfStructureElement(PdfTagType.TableDataCell);
//Draw the PdfGrid
pdfGrid.Draw(pdfPage, PointF.Empty);
//save the document and dispose it
pdfDocument.Save("Output.pdf");
pdfDocument.Close(true);
```

VB.NET

```
'Creates a new PDF document
Dim pdfDocument As PdfDocument = New PdfDocument()
pdfDocument.DocumentInformation.Title = "Table"
'Adds new page
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
'Initialize the new structure element with tag type table
Dim element As PdfStructureElement = New
PdfStructureElement(PdfTagType.Table)
'Create a new PdfGrid
Dim pdfGrid As PdfGrid = New PdfGrid()
'Adding tag to PDF grid
pdfGrid.PdfTag = element
'Add three columns
pdfGrid.Columns.Add(3)
'Add header.
pdfGrid.Headers.Add(1)
Dim pdfGridHeader As PdfGridRow = pdfGrid.Headers(0)
pdfGridHeader.Style.Font = New PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Bold)
pdfGridHeader.Style.TextBrush = PdfBrushes.Brown
'Adding tag for each row with tag type TR
pdfGridHeader.PdfTag = New PdfStructureElement(PdfTagType.TableRow)
pdfGridHeader.Cells(0).Value = "Employee ID"
'Adding tag for header cell with tag type TH
pdfGridHeader.Cells(0).PdfTag = New
PdfStructureElement(PdfTagType.TableHeader)
pdfGridHeader.Cells(1).Value = "Employee Name"
'Adding tag for header cell with tag type TH
pdfGridHeader.Cells(1).PdfTag = New
PdfStructureElement(PdfTagType.TableHeader)
pdfGridHeader.Cells(2).Value = "Salary"
'Adding tag for header cell with tag type TH
pdfGridHeader.Cells(2).PdfTag = New
PdfStructureElement(PdfTagType.TableHeader)
'Add rows.
Dim pdfGridRow As PdfGridRow = pdfGrid.Rows.Add()
pdfGridRow.PdfTag = New PdfStructureElement(PdfTagType.TableRow)
pdfGridRow.Cells(0).Value = "E01"
pdfGridRow.Cells(1).Value = "Clay"
pdfGridRow.Cells(2).Value = "$10,000"
'Adding tag for each cell with tag type TD
pdfGridRow.Cells(0).PdfTag = New
PdfStructureElement(PdfTagType.TableDataCell)
```

```
pdfGridRow.Cells(1).PdfTag = New
PdfStructureElement(PdfTagType.TableDataCell)
pdfGridRow.Cells(2).PdfTag = New
PdfStructureElement(PdfTagType.TableDataCell)
'Draw the PdfGrid
pdfGrid.Draw(pdfPage, PointF.Empty)
'save the document and dispose it
pdfDocument.Save("Output.pdf")
pdfDocument.Close(True)
```

UWP

```
//Creates a new PDF document
PdfDocument pdfDocument = new PdfDocument();
pdfDocument.DocumentInformation.Title = "Table";
//Adds new page
PdfPage pdfPage = pdfDocument.Pages.Add();
//Initialize the new structure element with tag type table
PdfStructureElement element = new PdfStructureElement(PdfTagType.Table);
//Create a new PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Adding tag to PDF grid
pdfGrid.PdfTag = element;
//Add three columns
pdfGrid.Columns.Add(3);
//Add header
pdfGrid.Headers.Add(1);
PdfGridRow pdfGridHeader = pdfGrid.Headers[0];
pdfGridHeader.Style.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Bold);
pdfGridHeader.Style.TextBrush = PdfBrushes.Brown;
//Adding tag for each row with tag type TR
pdfGridHeader.PdfTag = new PdfStructureElement(PdfTagType.TableRow);
pdfGridHeader.Cells[0].Value = "Employee ID";
//Adding tag for header cell with tag type TH
pdfGridHeader.Cells[0].PdfTag = new
PdfStructureElement(PdfTagType.TableHeader);
pdfGridHeader.Cells[1].Value = "Employee Name";
//Adding tag for header cell with tag type TH
pdfGridHeader.Cells[1].PdfTag = new
PdfStructureElement(PdfTagType.TableHeader);
pdfGridHeader.Cells[2].Value = "Salary";
//Adding tag for header cell with tag type TH
pdfGridHeader.Cells[2].PdfTag = new
PdfStructureElement(PdfTagType.TableHeader);
//Add rows
PdfGridRow pdfGridRow = pdfGrid.Rows.Add();
pdfGridRow.PdfTag = new PdfStructureElement(PdfTagType.TableRow);
pdfGridRow.Cells[0].Value = "E01";
pdfGridRow.Cells[1].Value = "Clay";
pdfGridRow.Cells[2].Value = "$10,000";
//Adding tag for each cell with tag type TD
pdfGridRow.Cells[0].PdfTag = new
PdfStructureElement(PdfTagType.TableDataCell);
pdfGridRow.Cells[1].PdfTag = new
PdfStructureElement(PdfTagType.TableDataCell);
```



```
pdfGridRow.Cells[2].PdfTag = new
PdfStructureElement(PdfTagType.TableDataCell);
//Draw the PdfGrid
pdfGrid.Draw(pdfPage, PointF.Empty);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
//Close the document
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for the respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Creates a new PDF document
PdfDocument pdfDocument = new PdfDocument();
pdfDocument.DocumentInformation.Title = "Table";
//Adds new page
PdfPage pdfPage = pdfDocument.Pages.Add();
//Initialize the new structure element with tag type table
PdfStructureElement element = new PdfStructureElement(PdfTagType.Table);
//Create a new PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Adding tag to PDF grid
pdfGrid.PdfTag = element;
//Add three columns
pdfGrid.Columns.Add(3);
//Add header
pdfGrid.Headers.Add(1);
PdfGridRow pdfGridHeader = pdfGrid.Headers[0];
pdfGridHeader.Style.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Bold);
pdfGridHeader.Style.TextBrush = PdfBrushes.Brown;
//Adding tag for each row with tag type TR
pdfGridHeader.PdfTag = new PdfStructureElement(PdfTagType.TableRow);
pdfGridHeader.Cells[0].Value = "Employee ID";
//Adding tag for header cell with tag type TH
pdfGridHeader.Cells[0].PdfTag = new
PdfStructureElement(PdfTagType.TableHeader);
pdfGridHeader.Cells[1].Value = "Employee Name";
//Adding tag for header cell with tag type TH
pdfGridHeader.Cells[1].PdfTag = new
PdfStructureElement(PdfTagType.TableHeader);
pdfGridHeader.Cells[2].Value = "Salary";
//Adding tag for header cell with tag type TH
pdfGridHeader.Cells[2].PdfTag = new
PdfStructureElement(PdfTagType.TableHeader);
//Add rows
PdfGridRow pdfGridRow = pdfGrid.Rows.Add();
pdfGridRow.PdfTag = new PdfStructureElement(PdfTagType.TableRow);
pdfGridRow.Cells[0].Value = "E01";
pdfGridRow.Cells[1].Value = "Clay";
pdfGridRow.Cells[2].Value = "$10,000";
//Adding tag for each cell with tag type TD
```

```

pdfGridRow.Cells[0].PdfTag = new
PdfStructureElement(PdfTagType.TableDataCell);
pdfGridRow.Cells[1].PdfTag = new
PdfStructureElement(PdfTagType.TableDataCell);
pdfGridRow.Cells[2].PdfTag = new
PdfStructureElement(PdfTagType.TableDataCell);
//Draw the PdfGrid
pdfGrid.Draw(pdfPage, PointF.Empty);
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Closes the document
pdfDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates a new PDF document
PdfDocument pdfDocument = new PdfDocument();
pdfDocument.DocumentInformation.Title = "Table";
//Adds new page
PdfPage pdfPage = pdfDocument.Pages.Add();
//Initialize the new structure element with tag type table
PdfStructureElement element = new PdfStructureElement(PdfTagType.Table);
//Create a new PdfGrid
PdfGrid pdfGrid = new PdfGrid();
//Adding tag to PDF grid
pdfGrid.PdfTag = element;
//Add three columns.
pdfGrid.Columns.Add(3);
//Add header.
pdfGrid.Headers.Add(1);
PdfGridRow pdfGridHeader = pdfGrid.Headers[0];
pdfGridHeader.Style.Font = new PdfStandardFont(PdfFontFamily.Helvetica, 12,
PdfFontStyle.Bold);
pdfGridHeader.Style.TextBrush = PdfBrushes.Brown;
//Adding tag for each row with tag type TR
pdfGridHeader.PdfTag = new PdfStructureElement(PdfTagType.TableRow);
pdfGridHeader.Cells[0].Value = "Employee ID";
//Adding tag for header cell with tag type TH
pdfGridHeader.Cells[0].PdfTag = new
PdfStructureElement(PdfTagType.TableHeader);
pdfGridHeader.Cells[1].Value = "Employee Name";
//Adding tag for header cell with tag type TH
pdfGridHeader.Cells[1].PdfTag = new
PdfStructureElement(PdfTagType.TableHeader);
pdfGridHeader.Cells[2].Value = "Salary";
//Adding tag for header cell with tag type TH

```

```

pdfGridHeader.Cells[2].PdfTag = new
PdfStructureElement(PdfTagType.TableHeader);
//Add rows
PdfGridRow pdfGridRow = pdfGrid.Rows.Add();
pdfGridRow.PdfTag = new PdfStructureElement(PdfTagType.TableRow);
pdfGridRow.Cells[0].Value = "E01";
pdfGridRow.Cells[1].Value = "Clay";
pdfGridRow.Cells[2].Value = "$10,000";
//Adding tag for each cell with tag type TD
pdfGridRow.Cells[0].PdfTag = new
PdfStructureElement(PdfTagType.TableDataCell);
pdfGridRow.Cells[1].PdfTag = new
PdfStructureElement(PdfTagType.TableDataCell);
pdfGridRow.Cells[2].PdfTag = new
PdfStructureElement(PdfTagType.TableDataCell);
//Draw the PdfGrid
pdfGrid.Draw(pdfPage, PointF.Empty);
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Close the document
pdfDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Adding tag to List Element

You can add the tags to list element in PDF document by specifying the tag type as **List** of [PdfTagType](#) Enum available in the [PdfStructureElement](#) class.

The following code example illustrates how to add tag support for list element.

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Sets document title
document.DocumentInformation.Title = "List";
//Add a new page to the document
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
SizeF size = page.Graphics.ClientSize;
//Create font

```

```

PdfFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 10,
PdfFontStyle.Italic);
graphics.DrawString("List:", new PdfStandardFont(PdfFontFamily.Helvetica,
12, PdfFontStyle.Bold), PdfBrushes.Blue, new Point(10, 0));
string[] products = { "Tools", "Grid", "Chart", "Edit", "Diagram", "XlsIO",
"Grouping", "Calculate", "PDF", "HTMLUI", "DocIO" };
//Create string format
PdfStringFormat format = new PdfStringFormat();
format.LineSpacing = 10f;
//Initialize new structure element with tag type List.
PdfStructureElement listElement = new PdfStructureElement(PdfTagType.List);
//Create ordered list
PdfOrderedList pdfList = new PdfOrderedList();
//Adding tag for list element
pdfList.PdfTag = listElement;
pdfList.Marker.Brush = PdfBrushes.Black;
pdfList.Indent = 20;
//Set format for sub list
pdfList.Font = font;
pdfList.StringFormat = format;
for (int i = 0; i < products.Length; i++)
{
pdfList.Items.Add(string.Concat("Essential ", products[i]));
//Adding tag for the list item
pdfList.Items[i].PdfTag = new PdfStructureElement(PdfTagType.ListItem);
}
//Draw the list
pdfList.Draw(page, new RectangleF(0, 20, size.Width, size.Height));
//Save and close the document
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument()
'Sets document title
document.DocumentInformation.Title = "List"
'Add a new page to the document
Dim page As PdfPage = document.Pages.Add()
Dim graphics As PdfGraphics = page.Graphics
Dim size As.SizeF = page.Graphics.ClientSize
'Create font
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.TimesRoman, 10,
PdfFontStyle.Italic)
graphics.DrawString("List:", New PdfStandardFont(PdfFontFamily.Helvetica,
12, PdfFontStyle.Bold), PdfBrushes.Blue, New Point(10, 0))
Dim products() As String = {"Tools", "Grid", "Chart", "Edit", "Diagram",
"XlsIO", "Grouping", "Calculate", "PDF", "HTMLUI", "DocIO"}
'Create string format
Dim format As PdfStringFormat = New PdfStringFormat()
format.LineSpacing = 10.0F
'Initialize new structure element with tag type list
Dim listElement As PdfStructureElement = New
PdfStructureElement(PdfTagType.List)
'Create ordered list

```

```

Dim pdfList As PdfOrderedList = New PdfOrderedList()
'Adding tag for list element
pdfList.PdfTag = listElement
pdfList.Marker.Brush = PdfBrushes.Black
pdfList.Indent = 20
'Set format for sub list
pdfList.Font = font
pdfList.StringFormat = format
For i As Integer = 0 To products.Length - 1
pdfList.Items.Add(String.Concat("Essential ", products(i)))
'Adding tag for the list item
pdfList.Items(i).PdfTag = New PdfStructureElement(PdfTagType.ListItem)
Next
'Draw the list
pdfList.Draw(page, New RectangleF(0, 20, size.Width, size.Height))
' Save and close the document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Sets document title
document.DocumentInformation.Title = "List";
//Add a new page to the document
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
SizeF size = page.Graphics.ClientSize;
//Create font
PdfFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 10,
PdfFontStyle.Italic);
graphics.DrawString("List:", new PdfStandardFont(PdfFontFamily.Helvetica,
12, PdfFontStyle.Bold), PdfBrushes.Blue, new PointF(10, 0));
string[] products = { "Tools", "Grid", "Chart", "Edit", "Diagram", "XlsIO",
"Grouping", "Calculate", "PDF", "HTMLUI", "DocIO" };
//Create string format
PdfStringFormat format = new PdfStringFormat();
format.LineSpacing = 10f;
//Initialize new structure element with tag type list
PdfStructureElement listElement = new PdfStructureElement(PdfTagType.List);
//Create Ordered list
PdfOrderedList pdfList = new PdfOrderedList();
//Adding tag for list element
pdfList.PdfTag = listElement;
pdfList.Marker.Brush = PdfBrushes.Black;
pdfList.Indent = 20;
//Set format for sub list
pdfList.Font = font;
pdfList.StringFormat = format;
for (int i = 0; i < products.Length; i++)
{
pdfList.Items.Add(string.Concat("Essential ", products[i]));
//Adding tag for the list item
pdfList.Items[i].PdfTag = new PdfStructureElement(PdfTagType.ListItem);
}

```

```
//Draw the list
pdfList.Draw(page, new RectangleF(0, 20, size.Width, size.Height));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Sets document title
document.DocumentInformation.Title = "List";
//Add a new page to the document
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
SizeF size = page.Graphics.ClientSize;
//Create font
PdfFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 10,
PdfFontStyle.Italic);
graphics.DrawString("List:", new PdfStandardFont(PdfFontFamily.Helvetica,
12, PdfFontStyle.Bold), PdfBrushes.Blue, new PointF(10, 0));
string[] products = { "Tools", "Grid", "Chart", "Edit", "Diagram", "XlsIO",
"Grouping", "Calculate", "PDF", "HTMLUI", "DocIO" };
//Create string format
PdfStringFormat format = new PdfStringFormat();
format.LineSpacing = 10f;
//Initialize new structure element with tag type List.
PdfStructureElement listElement = new PdfStructureElement(PdfTagType.List);
//Create ordered list
PdfOrderedList pdfList = new PdfOrderedList();
//Adding tag for list element
pdfList.PdfTag = listElement;
pdfList.Marker.Brush = PdfBrushes.Black;
pdfList.Indent = 20;
//Set format for sub list
pdfList.Font = font;
pdfList.StringFormat = format;
for (int i = 0; i < products.Length; i++)
{
pdfList.Items.Add(string.Concat("Essential ", products[i]));
//Adding tag for the list item
pdfList.Items[i].PdfTag = new PdfStructureElement(PdfTagType.ListItem);
}
//Draw the list
pdfList.Draw(page, new RectangleF(0, 20, size.Width, size.Height));
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
```

```
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Sets document title
document.DocumentInformation.Title = "List";
//Add a new page to the document
PdfPage page = document.Pages.Add();
PdfGraphics graphics = page.Graphics;
SizeF size = page.Graphics.ClientSize;
//Create font
PdfFont font = new PdfStandardFont(PdfFontFamily.TimesRoman, 10,
PdfFontStyle.Italic);
graphics.DrawString("List:", new PdfStandardFont(PdfFontFamily.Helvetica,
12, PdfFontStyle.Bold), PdfBrushes.Blue, new PointF(10, 0));
string[] products = { "Tools", "Grid", "Chart", "Edit", "Diagram", "XlsIO",
"Grouping", "Calculate", "PDF", "HTMLUI", "DocIO" };
//Create string format
PdfStringFormat format = new PdfStringFormat();
format.LineSpacing = 10f;
//Initialize new structure element with tag type List.
PdfStructureElement listElement = new PdfStructureElement(PdfTagType.List);
//Create ordered list
PdfOrderedList pdfList = new PdfOrderedList();
//Adding tag for list element
pdfList.PdfTag = listElement;
pdfList.Marker.Brush = PdfBrushes.Black;
pdfList.Indent = 20;
//Set format for sub list
pdfList.Font = font;
pdfList.StringFormat = format;
for (int i = 0; i < products.Length; i++)
{
pdfList.Items.Add(string.Concat("Essential ", products[i]));
//Adding tag for the list item
pdfList.Items[i].PdfTag = new PdfStructureElement(PdfTagType.ListItem);
}
//Draw the list
pdfList.Draw(page, new RectangleF(0, 20, size.Width, size.Height));
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Marking PDF content as an artifact

Artifacts in the PDF document can be graphic objects or other markings that are not a part of the authored content and will include such things as: headers, footers, page numbers, watermarks, cut marks, color bars, background images, lines separating content, or decorative images.

You can add artifact tag to PDF element by using the [PdfArtifact](#) class. The artifact type can be specified by using the [ArtifactType](#) property available in the [PdfArtifact](#) class.

The following code explains how to add tag for header and footers in the PDF document.

C#

```

//Creates new PDF document
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
pdfDocument.DocumentInformation.Title = "HeaderFooter";
//Creating artifact type for the header
PdfArtifact headerArtifact = new PdfArtifact(PdfArtifactType.Pagination, new
RectangleF(30, 40, 100, 100), new PdfAttached(PdfEdge.Top),
PdfArtifactSubType.Header);
//Create a header and draw the image
RectangleF bounds = new RectangleF(0, 0,
pdfDocument.Pages[0].GetClientSize().Width, 50);
PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
//Adding artifact to the header
header.PdfTag = headerArtifact;
PdfImage image = new PdfBitmap("syncfusion.jpg");
//Draw the image in the header
header.Graphics.DrawImage(image, new PointF(200, 0), new SizeF(100, 50));
//Add the header at the top
pdfDocument.Template.Top = header;
//Creating artifact type for the footer
PdfArtifact footerArtifact = new PdfArtifact(PdfArtifactType.Pagination, new
PdfAttached(PdfEdge.Bottom), PdfArtifactSubType.Footer);
//Create a Page template that can be used as footer
PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds);
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 7);
PdfBrush brush = new PdfSolidBrush(Color.Black);
//Create page number field
PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
//Create page count field
PdfPageCountField count = new PdfPageCountField(font, brush);
//Add the fields in composite fields

```



```

PdfCompositeField compositeField = new PdfCompositeField(font, brush, "Page
{0} of {1}", pageNumber, count);
compositeField.Bounds = footer.Bounds;
//Adding artifact type to the footer
compositeField.PdfTag = footerArtifact;
//Draw the composite field in footer
compositeField.Draw(footer.Graphics, new PointF(470, 40));
//Add the footer template at the bottom
pdfDocument.Template.Bottom = footer;
//Save the document and dispose it
pdfDocument.Save("HeaderFooter.pdf");
pdfDocument.Close(true);

```

VB.NET

```

'Creates new PDF document
Dim pdfDocument As PdfDocument = New PdfDocument()
'Add a page to the PDF document
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
pdfDocument.DocumentInformation.Title = "HeaderFooter"
'Creating artifact type for the header
Dim headerArtifact As PdfArtifact = New
PdfArtifact(PdfArtifactType.Pagination, New RectangleF(30, 40, 100, 100),
New PdfAttached(PdfEdge.Top), PdfArtifactSubType.Header)
'Create a header and draw the image
Dim bounds As RectangleF = New RectangleF(0, 0,
pdfDocument.Pages(0).GetClientSize().Width, 50)
Dim header As PdfPageTemplateElement = New PdfPageTemplateElement(bounds)
'Adding artifact to the header
header.PdfTag = headerArtifact
Dim image As PdfImage = New PdfBitmap("syncfusion.jpg")
'Draw the image in the header
header.Graphics.DrawImage(image, New PointF(200, 0), New SizeF(100, 50))
'Add the header at the top
pdfDocument.Template.Top = header
'Creating artifact type for the footer
Dim footerArtifact As PdfArtifact = New
PdfArtifact(PdfArtifactType.Pagination, New PdfAttached(PdfEdge.Bottom),
PdfArtifactSubType.Footer)
'Create a page template that can be used as footer
Dim footer As PdfPageTemplateElement = New PdfPageTemplateElement(bounds)
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 7)
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
'Create page number field
Dim pageNumber As PdfPageNumberField = New PdfPageNumberField(font, brush)
'Create page count field
Dim count As PdfPageCountField = New PdfPageCountField(font, brush)
'Add the fields in composite fields
Dim compositeField As PdfCompositeField = New PdfCompositeField(font, brush,
"Page {0} of {1}", pageNumber, count)
compositeField.Bounds = footer.Bounds
'Adding artifact type to the footer
compositeField.PdfTag = footerArtifact
'Draw the composite field in footer
compositeField.Draw(footer.Graphics, New PointF(470, 40))
'Add the footer template at the bottom

```

```
pdfDocument.Template.Bottom = footer
'Save the document and dispose it
pdfDocument.Save("HeaderFooter.pdf")
pdfDocument.Close(True)
```

UWP

```
//Creates new PDF document
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
pdfDocument.DocumentInformation.Title = "HeaderFooter";
//Creating artifact type for the header
PdfArtifact headerArtifact = new PdfArtifact(PdfArtifactType.Pagination, new
RectangleF(30, 40, 100, 100), new PdfAttached(PdfEdge.Top
//Create a header and draw the image
RectangleF bounds = new RectangleF(0, 0,
pdfDocument.Pages[0].GetClientSize().Width, 50);
PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
//Adding artifact to the header
header.PdfTag = headerArtifact;
//Load the image as stream
Stream imageStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.syncfusion.jpg");
PdfImage image = new PdfBitmap(imageStream);
//Draw the image in the header
header.Graphics.DrawImage(image, new PointF(200, 0), new SizeF(100, 50));
//Add the header at the top
pdfDocument.Template.Top = header;
//Creating artifact type for the footer
PdfArtifact footerArtifact = new PdfArtifact(PdfArtifactType.Pagination, new
PdfAttached(PdfEdge.Bottom), PdfArtifactSubType.Footer);
//Create a Page template that can be used as footer
PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds);
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 7);
PdfBrush brush = new PdfSolidBrush(Color.FromArgb(0, 0, 0, 0));
//Create page number field
PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
//Create page count field
PdfPageCountField count = new PdfPageCountField(font, brush);
//Add the fields in composite fields
PdfCompositeField compositeField = new PdfCompositeField(font, brush, "Page
{0} of {1}", pageNumber, count);
compositeField.Bounds = footer.Bounds;
//Adding artifact type to the footer
compositeField.PdfTag = footerArtifact;
//Draw the composite field in footer
compositeField.Draw(footer.Graphics, new PointF(470, 40));
//Add the footer template at the bottom
pdfDocument.Template.Bottom = footer;
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
//Close the document
pdfDocument.Close(true);
```

```
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for the respective code samples
Save(stream, "HeaderFooter.pdf");
```

ASP.NET CORE

```
//Creates new PDF document
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
pdfDocument.DocumentInformation.Title = "HeaderFooter";
//Creating artifact type for the header
PdfArtifact headerArtifact = new PdfArtifact(PdfArtifactType.Pagination, new
RectangleF(30, 40, 100, 100), new PdfAttached(PdfEdge.Top),
PdfArtifactSubType.Header);
//Create a header and draw the image
RectangleF bounds = new RectangleF(0, 0,
pdfDocument.Pages[0].GetClientSize().Width, 50);
PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
//Adding artifact to the header
header.PdfTag = headerArtifact;
//Load the image as stream
FileStream imageStream = new FileStream("Autumn Leaves.jpg", FileMode.Open,
FileAccess.Read);
PdfImage image = new PdfBitmap(imageStream);
//Draw the image in the header
header.Graphics.DrawImage(image, new PointF(200, 0), new SizeF(100, 50));
//Add the header at the top
pdfDocument.Template.Top = header;
//Creating artifact type for the footer
PdfArtifact footerArtifact = new PdfArtifact(PdfArtifactType.Pagination, new
PdfAttached(PdfEdge.Bottom), PdfArtifactSubType.Footer);
//Create a Page template that can be used as footer
PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds);
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 7);
PdfBrush brush = new PdfSolidBrush(Color.Black);
//Create page number field
PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
//Create page count field
PdfPageCountField count = new PdfPageCountField(font, brush);
//Add the fields in composite fields
PdfCompositeField compositeField = new PdfCompositeField(font, brush, "Page
{0} of {1}", pageNumber, count);
compositeField.Bounds = footer.Bounds;
//Adding artifact type to the footer
compositeField.PdfTag = footerArtifact;
//Draw the composite field in footer
compositeField.Draw(footer.Graphics, new PointF(470, 40));
//Add the footer template at the bottom
pdfDocument.Template.Bottom = footer;
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Closes the document
pdfDocument.Close(true);
```

```
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "HeaderFooter.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates new PDF document
PdfDocument pdfDocument = new PdfDocument();
//Add a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
pdfDocument.DocumentInformation.Title = "HeaderFooter";
//Creating artifact type for the header
PdfArtifact headerArtifact = new PdfArtifact(PdfArtifactType.Pagination, new
RectangleF(30, 40, 100, 100), new PdfAttached(PdfEdge.Top),
PdfArtifactSubType.Header);
//Create a header and draw the image
RectangleF bounds = new RectangleF(0, 0,
pdfDocument.Pages[0].GetClientSize().Width, 50);
PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
//Adding artifact to the header
header.PdfTag = headerArtifact;
//Load the file as stream
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Signature.Asse
ts.Autumn Leaves.jpg");
PdfImage image = new PdfBitmap(imageStream);
//Draw the image in the header
header.Graphics.DrawImage(image, new PointF(200, 0), new SizeF(100, 50));
//Add the header at the top
pdfDocument.Template.Top = header;
//Creating artifact type for the footer
PdfArtifact footerArtifact = new PdfArtifact(PdfArtifactType.Pagination, new
PdfAttached(PdfEdge.Bottom), PdfArtifactSubType.Footer);
//Create a Page template that can be used as footer
PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds);
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 7);
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Black);
//Create page number field
PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
//Create page count field
PdfPageCountField count = new PdfPageCountField(font, brush);
//Add the fields in composite fields
PdfCompositeField compositeField = new PdfCompositeField(font, brush, "Page
{0} of {1}", pageNumber, count);
compositeField.Bounds = footer.Bounds;
//Adding artifact type to the footer
compositeField.PdfTag = footerArtifact;
//Draw the composite field in footer
compositeField.Draw(footer.Graphics, new PointF(470, 40));
//Add the footer template at the bottom
pdfDocument.Template.Bottom = footer;
//Save the document into stream
```

```

MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Close the document.
pdfDocument.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("HeaderFooter.
pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("HeaderFooter.pdf",
"application/pdf", stream);
}

```

Tag Reading Order

Basically, the element which draws first takes precedence over the tag reading order. You can re-order the tagged elements in document using the [Order](#) property.

The following code example illustrates how to order the tagged elements in a PDF document.

C#

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Sets document title
document.DocumentInformation.Title = "Order";
//Add a new page to the document
PdfPage page = document.Pages.Add();
//Initialize the structure element with tag type paragraph
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Paragraph);
//Order the tag in third position
structureElement.Order = 3;
PdfTextElement element = new PdfTextElement("This is paragraph ONE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Adding tag to the text element
element.PdfTag = structureElement;
element.Draw(page, new RectangleF(0, 0, page.Graphics.ClientSize.Width / 2,
200));
//Initialize the structure element with tag type paragraph
PdfStructureElement paraStruct1 = new
PdfStructureElement(PdfTagType.Paragraph);
//Order the tag in first position
paraStruct1.Order = 1;
//Creates new text element
PdfTextElement element1 = new PdfTextElement("This is paragraph TWO.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element1.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Adding tag to the text element

```

```

element1.PdfTag = paraStruct1;
element1.Draw(page, new RectangleF(0, 50, page.Graphics.ClientSize.Width /
2, 200));
//Initialize the structure element with tag type paragraph
PdfStructureElement paraStruct2 = new
PdfStructureElement(PdfTagType.Paragraph);
//Order the tag in second position
paraStruct2.Order = 2;
//Creates new text element
PdfTextElement element2 = new PdfTextElement("This is paragraph THREE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element2.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Adding tag to the text element
element2.PdfTag = paraStruct2;
element2.Draw(page.Graphics, new PointF(0, 100));
//Save the document and dispose it
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim document As PdfDocument = New PdfDocument()
'Sets document title
document.DocumentInformation.Title = "Order"
'Add a new page to the document
Dim page As PdfPage = document.Pages.Add()
'Initialize the structure element with tag type paragraph.
Dim structureElement As PdfStructureElement = New
PdfStructureElement(PdfTagType.Paragraph)
'Order the tag in third position
structureElement.Order = 3
Dim element As PdfTextElement = New PdfTextElement("This is paragraph ONE.",
New PdfStandardFont(PdfFontFamily.Helvetica, 12))
element.Brush = New PdfSolidBrush(New PdfColor(89, 89, 93))
'Adding tag to the text element
element.PdfTag = structureElement
element.Draw(page, New RectangleF(0, 0, page.Graphics.ClientSize.Width / 2,
200))
'Initialize the structure element with tag type paragraph
Dim paraStruct1 As PdfStructureElement = New
PdfStructureElement(PdfTagType.Paragraph)
'Order the tag in first position
paraStruct1.Order = 1
'Creates new text element
Dim element1 As PdfTextElement = New PdfTextElement("This is paragraph
TWO.", New PdfStandardFont(PdfFontFamily.Helvetica, 12))
element1.Brush = New PdfSolidBrush(New PdfColor(89, 89, 93))
'Adding tag to the text element
element1.PdfTag = paraStruct1
element1.Draw(page, New RectangleF(0, 50, page.Graphics.ClientSize.Width /
2, 200))
'Initialize the structure element with tag type paragraph
Dim paraStruct2 As PdfStructureElement = New
PdfStructureElement(PdfTagType.Paragraph)
'Order the tag in second position

```

```

paraStruct2.Order = 2
'Creates new text element
Dim element2 As PdfTextElement = New PdfTextElement("This is paragraph
THREE.", New PdfStandardFont(PdfFontFamily.Helvetica, 12))
element2.Brush = New PdfSolidBrush(New PdfColor(89, 89, 93))
'Adding tag to the text element
element2.PdfTag = paraStruct2
element2.Draw(page.Graphics, New PointF(0, 100))
'Save the document and dispose it
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Sets document title
document.DocumentInformation.Title = "Order";
//Add a new page to the document
PdfPage page = document.Pages.Add();
//Initialize the structure element with tag type paragraph
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Paragraph);
//Order the tag in third position
structureElement.Order = 3;
PdfTextElement element = new PdfTextElement("This is paragraph ONE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Adding tag to the text element
element.PdfTag = structureElement;
element.Draw(page, new RectangleF(0, 0, page.Graphics.ClientSize.Width / 2,
200));
//Initialize the structure element with tag type paragraph.
PdfStructureElement paraStruct1 = new
PdfStructureElement(PdfTagType.Paragraph);
//Order the tag in first position
paraStruct1.Order = 1;
//Creates new text element
PdfTextElement element1 = new PdfTextElement("This is paragraph TWO.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element1.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Adding tag to the text element
element1.PdfTag = paraStruct1;
element1.Draw(page, new RectangleF(0, 50, page.Graphics.ClientSize.Width /
2, 200));
//Initialize the structure element with tag type paragraph
PdfStructureElement paraStruct2 = new
PdfStructureElement(PdfTagType.Paragraph);
//Order the tag in second position
paraStruct2.Order = 2;
//Creates new text element
PdfTextElement element2 = new PdfTextElement("This is paragraph THREE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element2.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Adding tag to the text element
element2.PdfTag = paraStruct2;

```

```

element2.Draw(page.Graphics, new PointF(0, 100));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Sets document title
document.DocumentInformation.Title = "Order";
//Add a new page to the document
PdfPage page = document.Pages.Add();
//Initialize the structure element with tag type paragraph
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Paragraph);
//Order the tag in third position
structureElement.Order = 3;
PdfTextElement element = new PdfTextElement("This is paragraph ONE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Adding tag to the text element
element.PdfTag = structureElement;
element.Draw(page, new RectangleF(0, 0, page.Graphics.ClientSize.Width / 2,
200));
//Initialize the structure element with tag type paragraph
PdfStructureElement paraStruct1 = new
PdfStructureElement(PdfTagType.Paragraph);
//Order the tag in first position
paraStruct1.Order = 1;
//Creates new text element
PdfTextElement element1 = new PdfTextElement("This is paragraph TWO.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element1.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Adding tag to the text element
element1.PdfTag = paraStruct1;
element1.Draw(page, new RectangleF(0, 50, page.Graphics.ClientSize.Width /
2, 200));
//Initialize the structure element with tag type paragraph
PdfStructureElement paraStruct2 = new
PdfStructureElement(PdfTagType.Paragraph);
//Order the tag in second position
paraStruct2.Order = 2;
//Creates new text element
PdfTextElement element2 = new PdfTextElement("This is paragraph THREE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element2.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Adding tag to the text element
element2.PdfTag = paraStruct2;
element2.Draw(page.Graphics, new PointF(0, 100));
//Save the document into stream

```



```

MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new PDF document
PdfDocument document = new PdfDocument();
//Sets document title
document.DocumentInformation.Title = "Order";
//Add a new page to the document
PdfPage page = document.Pages.Add();
//Initialize the structure element with tag type paragraph
PdfStructureElement structureElement = new
PdfStructureElement(PdfTagType.Paragraph);
//Order the tag in third position
structureElement.Order = 3;
PdfTextElement element = new PdfTextElement("This is paragraph ONE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Adding tag to the text element
element.PdfTag = structureElement;
element.Draw(page, new RectangleF(0, 0, page.Graphics.ClientSize.Width / 2,
200));
//Initialize the structure element with tag type paragraph
PdfStructureElement paraStruct1 = new
PdfStructureElement(PdfTagType.Paragraph);
//Order the tag in first position
paraStruct1.Order = 1;
//Creates new text element
PdfTextElement element1 = new PdfTextElement("This is paragraph TWO.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element1.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Adding tag to the text element
element1.PdfTag = paraStruct1;
element1.Draw(page, new RectangleF(0, 50, page.Graphics.ClientSize.Width /
2, 200));
//Initialize the structure element with tag type paragraph
PdfStructureElement paraStruct2 = new
PdfStructureElement(PdfTagType.Paragraph);
//Order the tag in second position
paraStruct2.Order = 2;
//Creates new text element
PdfTextElement element2 = new PdfTextElement("This is paragraph THREE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element2.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
//Adding tag to the text element

```

```

element2.PdfTag = paraStruct2;
element2.Draw(page.Graphics, new PointF(0, 100));
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Auto Tagging a new document

When the auto-tag feature is enabled, all the elements in the document is tagged with appropriate tag type that is **Paragraph**, **Figure**, **Annotation**, and more from [PdfTagType](#) Enum.

The following code example explains how to auto-tag the elements in a PDF document.

Note: Enabling the auto-tag feature will never add alternate texts/descriptions for figures, images, and other properties related to tag.

C#

```

//Creates new PDF document
PdfDocument document = new PdfDocument();
//Set true to auto tag all elements in document
document.AutoTag = true;
document.DocumentInformation.Title = "AutoTag";
// Add a new page to the document
PdfPage page = document.Pages.Add();
//Creates new text element
PdfTextElement element = new PdfTextElement("This is paragraph ONE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
element.Draw(page, new RectangleF(0, 0, page.Graphics.ClientSize.Width / 2,
200));
//Creates new text element
PdfTextElement element1 = new PdfTextElement("This is paragraph TWO.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element1.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
element1.Draw(page, new RectangleF(0, 50, page.Graphics.ClientSize.Width /
2, 200));
//Creates new text element
PdfTextElement element2 = new PdfTextElement("This is paragraph THREE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));

```

```

element2.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
element2.Draw(page.Graphics, new PointF(0, 100));
//Save the document and dispose it
document.Save("AutoTag.pdf");
document.Close(true);

```

VB.NET

```

'Creates new PDF document
Dim document As PdfDocument = New PdfDocument()
'Set true to auto tag all elements in document
document.AutoTag = True
document.DocumentInformation.Title = "AutoTag"
'Add a new page to the document
Dim page As PdfPage = document.Pages.Add()
'Creates new text element
Dim element As PdfTextElement = New PdfTextElement("This is paragraph ONE.",
New PdfStandardFont(PdfFontFamily.Helvetica, 12))
element.Brush = New PdfSolidBrush(New PdfColor(89, 89, 93))
element.Draw(page, New RectangleF(0, 0, page.Graphics.ClientSize.Width / 2,
200))
'Creates new text element
Dim element1 As PdfTextElement = New PdfTextElement("This is paragraph
TWO.", New PdfStandardFont(PdfFontFamily.Helvetica, 12))
element1.Brush = New PdfSolidBrush(New PdfColor(89, 89, 93))
element1.Draw(page, New RectangleF(0, 50, page.Graphics.ClientSize.Width /
2, 200))
'Creates new text element
Dim element2 As PdfTextElement = New PdfTextElement("This is paragraph
THREE.", New PdfStandardFont(PdfFontFamily.Helvetica, 12))
element2.Brush = New PdfSolidBrush(New PdfColor(89, 89, 93))
element2.Draw(page.Graphics, New PointF(0, 100))
'Save the document and dispose it
document.Save("AutoTag.pdf")
document.Close(True)

```

UWP

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Set true to auto tag all elements in document
document.AutoTag = true;
document.DocumentInformation.Title = "AutoTag";
//Add a new page to the document
PdfPage page = document.Pages.Add();
//Creates new text element
PdfTextElement element = new PdfTextElement("This is paragraph ONE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
element.Draw(page, new RectangleF(0, 0, page.Graphics.ClientSize.Width / 2,
200));
//Creates new text element
PdfTextElement element1 = new PdfTextElement("This is paragraph TWO.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element1.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));

```

```

element1.Draw(page, new RectangleF(0, 50, page.Graphics.ClientSize.Width /
2, 200));
//Creates new text element
PdfTextElement element2 = new PdfTextElement("This is paragraph THREE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element2.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
element2.Draw(page.Graphics, new PointF(0, 100));
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "AutoTag.pdf");

```

ASP.NET CORE

```

//Creates new PDF document
PdfDocument document = new PdfDocument();
//Set true to auto tag all elements in document
document.AutoTag = true;
document.DocumentInformation.Title = "AutoTag";
// Add a new page to the document
PdfPage page = document.Pages.Add();
//Creates new text element
PdfTextElement element = new PdfTextElement("This is paragraph ONE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
element.Draw(page, new RectangleF(0, 0, page.Graphics.ClientSize.Width / 2,
200));
//Creates new text element
PdfTextElement element1 = new PdfTextElement("This is paragraph TWO.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element1.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
element1.Draw(page, new RectangleF(0, 50, page.Graphics.ClientSize.Width /
2, 200));
//Creates new text element
PdfTextElement element2 = new PdfTextElement("This is paragraph THREE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element2.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
element2.Draw(page.Graphics, new PointF(0, 100));
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "AutoTag.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates new PDF document
PdfDocument document = new PdfDocument();
//Set true to auto tag all elements in document
document.AutoTag = true;
document.DocumentInformation.Title = "AutoTag";
// Add a new page to the document
PdfPage page = document.Pages.Add();
//Creates new text element
PdfTextElement element = new PdfTextElement("This is paragraph ONE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
element.Draw(page, new RectangleF(0, 0, page.Graphics.ClientSize.Width / 2,
200));
//Creates new text element
PdfTextElement element1 = new PdfTextElement("This is paragraph TWO.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element1.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
element1.Draw(page, new RectangleF(0, 50, page.Graphics.ClientSize.Width /
2, 200));
//Creates new text element
PdfTextElement element2 = new PdfTextElement("This is paragraph THREE.", new
PdfStandardFont(PdfFontFamily.Helvetica, 12));
element2.Brush = new PdfSolidBrush(new PdfColor(89, 89, 93));
element2.Draw(page.Graphics, new PointF(0, 100));
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("AutoTag.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("AutoTag.pdf",
"application/pdf", stream);
}

```

Note: After the document is auto tagged and if any element tagged manually, then the manually tagged element takes the precedence.

How to pass accessibility full check

To pass the full check accessibility, follow the below conventions while tagging the document:

1. Mention the PDF document "Title" in the document properties.
2. Make sure that the images in the document has alternate text or marked as artifact.

3. The bookmarks should be included for tagged PDF with more than 21 pages.
4. All the form fields require text description (tooltip).
5. All tables in a document should have a header.
6. Tables must contain same number of columns in each row.
7. A list element must contain list item element (LI), and list item element can only contain label (Lb1) or body elements(LBody).

Tagged PDF support for converting HTML to PDF

Essential PDF provides support to convert HTML to TaggedPDF by using the MSHTML rendering library.

The Tagged PDF is a stylized use of PDF that builds the logical structure Framework. It defines a set of standard structure types and attributes that allows the page content (text, graphics, and images) to be extracted and reused. The contents are accessible to users with visual impairments.

To convert HTML to Tagged PDF, you can use the [ConvertToTaggedPDF](#) method in [HtmlConverter](#) class.

The following code illustrates how to convert HTML to TaggedPDF.

C#

```
//Creates a new PdfDocument
PdfDocument document = new PdfDocument();
//Creates a new instance of HtmlConverter class
using (HtmlConverter html = new HtmlConverter())
{
    //Enable JavaScript
    html.EnableJavaScript = true;
    //Converts to Tagged PDF
    html.ConvertToTaggedPDF(document, "http://www.google.com");
}
//Saves and closes the document
document.Save("Sample.pdf");
document.Close(true);
```

VB.NET

```
'Creates a new PdfDocument
Dim document As New PdfDocument()
'Creates a new instance of HtmlConverter class
Using html As New HtmlConverter()
    'Enables JavaScript
    html.EnableJavaScript = True
    'Converts to Tagged PDF
    html.ConvertToTaggedPDF(document, "http://www.google.com")
End Using
'Saves and closes the document
document.Save("Sample.pdf")
document.Close(True)
```

Note: Hyperlinks are not supported in tagged PDF

Converting Word document to Tagged PDF

This setting allows you to determine whether to preserve document structured tags in the converted PDF document for accessibility (508 compliance) support. This property will set the title and description

for images, diagrams, and other objects in the generated PDF document. This information will be useful for people with vision or cognitive impairments who cannot see or understand the object.

The following code sample shows how to preserve document structured tags in the converted PDF document.

C#

```
//Loads an existing Word document
WordDocument wordDocument = new WordDocument("Sample.docx",
FormatType.Docx);
//Creates an instance of the DocToPDFConverter - responsible for Word to PDF conversion
DocToPDFConverter converter = new DocToPDFConverter();
//Sets true to preserve document structured tags in the converted PDF document
converter.Settings.AutoTag = true;
//Converts Word document into PDF document
PdfDocument pdfDocument = converter.ConvertToPDF(wordDocument);
//Saves the PDF file to file system
pdfDocument.Save("WordtoPDF.pdf");
//Closes the instance of document objects
pdfDocument.Close(true);
wordDocument.Close();
```

VB.NET

```
'Loads an existing Word document
Dim wordDocument As New WordDocument("Sample.docx", FormatType.Docx)
'Creates an instance of the DocToPDFConverter - responsible for Word to PDF conversion
Dim converter As New DocToPDFConverter()
'Sets true to preserve document structured tags in the converted PDF document
converter.Settings.AutoTag = True
'Converts Word document into PDF document
Dim pdfDocument As PdfDocument = converter.ConvertToPDF(wordDocument)
'Saves the PDF file to file system
pdfDocument.Save("WordtoPDF.pdf")
'Closes the instance of document objects
pdfDocument.Close(True)
wordDocument.Close()
```

ASP.NET CORE

```
//Open the file as Stream
FileStream docStream = new FileStream(@"D:\Template.docx", FileMode.Open,
FileAccess.Read);
//Loads file stream into Word document
WordDocument wordDocument = new WordDocument(docStream,
Syncfusion.DocIO.FormatTypeAutomatic);
//Instantiation of DocIORenderer for Word to PDF conversion
DocIORenderer render = new DocIORenderer();
//Sets true to preserve document structured tags in the converted PDF document
render.Settings.AutoTag = true;
```

```
//Converts Word document into PDF document
PdfDocument pdfDocument = render.ConvertToPDF(wordDocument);
//Releases all resources used by the Word document and DocIO Renderer
objects
render.Dispose();
wordDocument.Dispose();
//Saves the PDF file
MemoryStream outputStream = new MemoryStream();
pdfDocument.Save(outputStream);
//Closes the instance of PDF document object
pdfDocument.Close();
```

XAMARIN

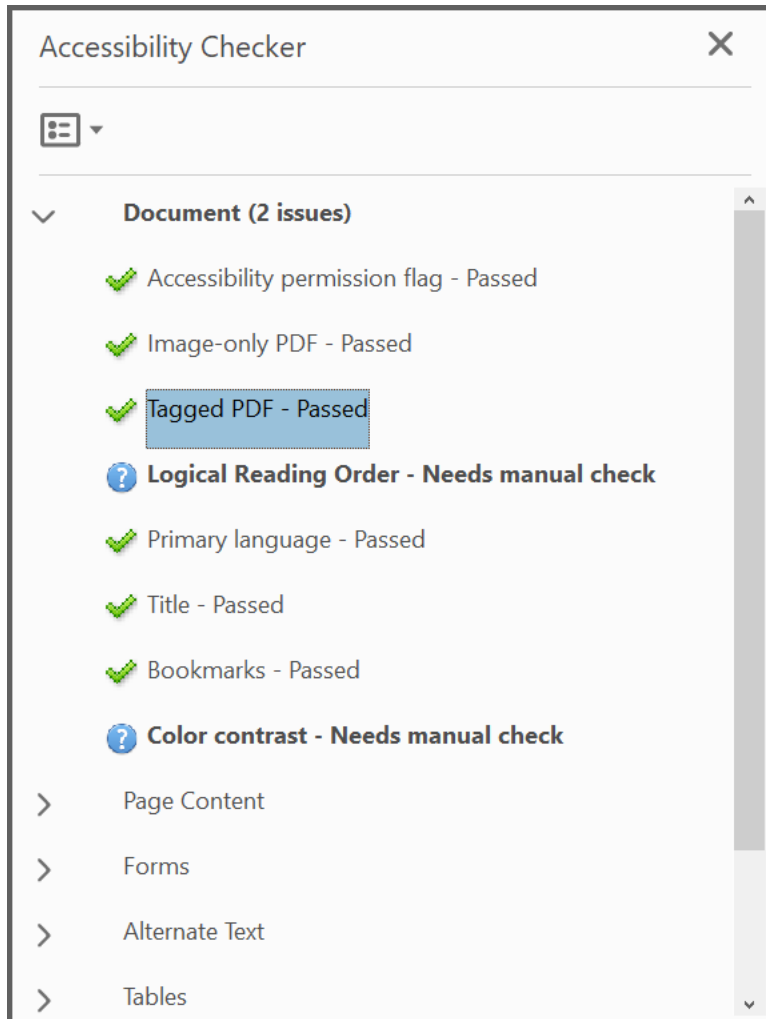
```
//Load the Word document as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.docx");
//Loads the stream into Word Document.
WordDocument wordDocument = new WordDocument(docStream,
Syncfusion.DocIO.FormatType.Automatic);
//Instantiation of DocIORenderer for Word to PDF conversion
DocIORenderer render = new DocIORenderer();
//Sets true to preserve document structured tags in the converted PDF
document
render.Settings.AutoTag = true;
//Converts Word document into PDF document
PdfDocument pdfDocument = render.ConvertToPDF(wordDocument);
//Releases all resources used by the Word document and DocIO Renderer
objects
render.Dispose();
wordDocument.Dispose();
//Saves the PDF file
MemoryStream outputStream = new MemoryStream();
pdfDocument.Save(outputStream);
//Closes the instance of PDF document object
pdfDocument.Close();
```

Validating tagged PDF in Acrobat

Follow the below steps to validate the tagged PDF information in Adobe Acrobat:

1. Choose Tools > Accessibility. 2. In the secondary toolbar, click Full Check. 3. The Accessibility Checker Option dialog box will be displayed, from that you can ensure the validity of tagged information.

The following screenshot shows the Accessibility checker dialog box.



Working with Compression

Essential PDF allows you to compress the PDF document and thereby reduce the file size in the following three ways.

1. Compress an existing PDF document
2. Content compression for a new document
3. Image compression for a new document

Note: PDF supports compressing PDF document only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.

Compressing existing PDF document

You can compress the existing PDF document by using [PdfLoadedDocument](#) and [PdfCompressionOptions](#). The following compression techniques are used to compress the existing PDF document.

1. Compress the image with image quality
2. Optimizing embedded font
3. Optimizing page content

4. Remove metadata information

Compressing images with image quality

You can compress all the images of an existing PDF document by enabling the [CompressImages](#) property and assigning [ImageQuality](#) available in [PdfCompressionOptions](#) class. The [ImageQuality](#) property is used to reduce the quality of the image based on percentage value, where 100 is unchanged quality and 10 is low quality.

The following example code snippet illustrates how to compress the images in existing PDF document.

C#

```
//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("input.pdf");
//Create a new compression option.
PdfCompressionOptions options = new PdfCompressionOptions();
//Enable the compress image.
options.CompressImages = true;
//Set the image quality.
options.ImageQuality = 50;
//Assign the compression option to the document
loadedDocument.CompressionOptions = options;
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load the existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("input.pdf")
'Create a new compression option.
Dim options As PdfCompressionOptions = New PdfCompressionOptions()
'Enable the compress image.
options.CompressImages = True
'Set the image quality.
options.ImageQuality = 50
'Assign the compression option to the document
loadedDocument.CompressionOptions = options
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the document
loadedDocument.Close(True)
```

Optimizing embedded font

You can optimize the embedded fonts in an existing PDF document by enabling the [OptimizeFont](#) property available in the [PdfCompressionOptions](#) class. This technique reduces the font file size by removing all the unused glyph data.

The following example code snippet illustrates how to optimize embedded font in existing PDF document.

C#

```
//Load the existing PDF document
```

```

PdfLoadedDocument loadedDocument = new PdfLoadedDocument("input.pdf");
//Create a new compression option.
PdfCompressionOptions options = new PdfCompressionOptions();
//Enable the optimize font option
options.OptimizeFont = true;
//Assign the compression option to the document
loadedDocument.CompressionOptions = options;
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("input.pdf")
'Create a new compression option.
Dim options As PdfCompressionOptions = New PdfCompressionOptions()
'Enable the optimize font option
options.OptimizeFont = True
'Assign the compression option to the document
loadedDocument.CompressionOptions = options
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the document
loadedDocument.Close(True)

```

Note: The font compression support only in TrueType and Type2 embedded fonts.

Optimizing page contents

You can compress the page contents in an existing PDF document by enabling the [OptimizePageContents](#) property available in the [PdfCompressionOptions](#) class. This technique removes the unwanted data in the content streams such as commented lines, white spaces, etc.,

The following example code snippet illustrates how to optimize page contents in existing PDF document.

C#

```

//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("input.pdf");
//Create a new compression option.
PdfCompressionOptions options = new PdfCompressionOptions();
//Enable the optimize page contents.
options.OptimizePageContents = true;
//Assign the compression option to the document
loadedDocument.CompressionOptions = options;
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("input.pdf")

```

```

'Create a new compression option.
Dim options As PdfCompressionOptions = New PdfCompressionOptions()
'Enable the optimize page contents.
options.OptimizePageContents = True
'Assign the compression option to the document
loadedDocument.CompressionOptions = options
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the document
loadedDocument.Close(True)

```

Remove metadata information

You can reduce the PDF file size by removing the PDF document metadata information. This can be achieved by enabling the [RemoveMetadata](#) property available in the [PdfCompressionOptions](#) class.

The following example code snippet illustrates how to optimize page contents in existing PDF document.

C#

```

//Load the existing PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("input.pdf");
//Create a new compression option.
PdfCompressionOptions options = new PdfCompressionOptions();
//Set to remove the metadata information.
options.RemoveMetadata = true;
//Assign the compression option to the document
loadedDocument.CompressionOptions = options;
//Save the PDF document
loadedDocument.Save("Output.pdf");
//Close the document
loadedDocument.Close(true);

```

VB.NET

```

'Load the existing PDF document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("input.pdf")
'Create a new compression option.
Dim options As PdfCompressionOptions = New PdfCompressionOptions()
'Set to remove the metadata information.
options.RemoveMetadata = True
'Assign the compression option to the document
loadedDocument.CompressionOptions = options
'Save the PDF document
loadedDocument.Save("Output.pdf")
'Close the document
loadedDocument.Close(True)

```

Compressing the PDF content

Essential PDF allows you to control the compression level of the document by using the [PdfCompressionLevel](#) Enum. The compression level can be set to best, normal, none etc...

Content compression involves,

- 1) Removing all extra space characters.
- 2) Inserting a single repeat character to indicate a string of repeated characters.
- 3) Substituting smaller bit strings for frequently occurring characters.

The following code snippet illustrates how to compress the content of the PDF document.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Set the compression level to best
document.Compression = PdfCompressionLevel.Best;
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Set the font.
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 20);
string text = "Hello World!!!";
PdfTextElement textElement = new PdfTextElement(text, font);
PdfLayoutResult result = textElement.Draw(page, new RectangleF(0, 0,
font.MeasureString(text).Width, page.GetClientSize().Height));
for (int i = 0; i < 1000; i++)
{
    result = textElement.Draw(result.Page, new RectangleF(0,
    result.Bounds.Bottom + 10, font.MeasureString(text).Width,
    page.GetClientSize().Height));
}
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Set the compression level to best
document.Compression = PdfCompressionLevel.Best
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Set the font.
Dim font As PdfFont = New PdfStandardFont(PdfFontFamily.Helvetica, 20)
Dim text As String = "Hello World!!!"
Dim textElement As New PdfTextElement(text, font)
Dim result As PdfLayoutResult = textElement.Draw(page, New RectangleF(0, 0,
font.MeasureString(text).Width, page.GetClientSize().Height))
For i As Integer = 0 To 999
    result = textElement.Draw(result.Page, New RectangleF(0,
    result.Bounds.Bottom + 10, font.MeasureString(text).Width,
    page.GetClientSize().Height))
Next
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)
```

You can compress the existing PDF document by using the following code snippet.

C#

```
//Load the PDF document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Disable the incremental update
loadedDocument.FileStructure.IncrementalUpdate = false;
//Set the compression level
loadedDocument.Compression = PdfCompressionLevel.Best;
//Save and close the document
loadedDocument.Save("Output.pdf");
loadedDocument.Close(true);
```

VB.NET

```
'Load the PDF document
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Disable the incremental update
loadedDocument.FileStructure.IncrementalUpdate = False
'Set the compression level
loadedDocument.Compression = PdfCompressionLevel.Best
'Save and close the document
loadedDocument.Save("Output.pdf")
loadedDocument.Close(True)
```

Compressing images

Essential PDF allows you to compress/change the quality of the image in the PDF document by using the following code snippet.

C#

```
//Create a new PDF document.
PdfDocument document = new PdfDocument();
//Add a page to the document.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
PdfBitmap image = new PdfBitmap("Input.jpg");
//Reduce the quality of the image
image.Quality = 50;
image.Draw(page, new PointF(0, 0));
//Save the document.
document.Save("Output.pdf");
//Close the document.
document.Close(true);
```

VB.NET

```
'Create a new PDF document.
Dim document As New PdfDocument()
'Add a page to the document.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
Dim image As New PdfBitmap("Input.jpg")
'Reduce the quality of the image
```

```

image.Quality = 50
image.Draw(page, New PointF(0, 0))
'Save the document.
document.Save("Output.pdf")
'Close the document.
document.Close(True)

```

You can compress the images in the existing PDF document by using the following code snippet.

C#

```

//Load the PDF document which c images
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Disable the incremental update
loadedDocument.FileStructure.IncrementalUpdate = false;
//iterate all the pages to replace images
foreach (PdfPageBase page in loadedDocument.Pages)
{
    //Extract the images from the document
    Image[] extractedImages = page.ExtractImages();
    //Iterate all the image
    for (int j = 0; j < extractedImages.Count(); j++)
    {
        PdfBitmap image = new PdfBitmap(extractedImages[j]);
        //reduce the quality of the image
        image.Quality = 50;
        //replace the compressed image with old image in the PDF document
        page.ReplaceImage(j, image);
    }
}
//Save and close the document
loadedDocument.Save("Output.pdf");
loadedDocument.Close(true);

```

VB.NET

```

'Load the PDF document which consist of images
Dim loadedDocument As New PdfLoadedDocument("Input.pdf")
'Disable the incremental update
loadedDocument.FileStructure.IncrementalUpdate = False
'iterate all the pages to replace images
For Each page As PdfPageBase In loadedDocument.Pages
    'Extract the images from the document
    Dim extractedImages As Image() = page.ExtractImages()
    'Iterate all the image
    For j As Integer = 0 To extractedImages.Count() - 1
        Dim image As New PdfBitmap(extractedImages(j))
        'reduce the quality of the image
        image.Quality = 50
        'replace the compressed image with old image in the PDF document
        page.ReplaceImage(j, image)
    Next
Next
'Save and close the document
loadedDocument.Save("Output.pdf")
loadedDocument.Close(True)

```

Working with PDF Conformance

The Essential PDF currently supports the following PDF conformances:

- PDF/A-1a conformance
- PDF/A-1b conformance
- PDF/X-1a conformance
- PDF/A-2a conformance
- PDF/A-2b conformance
- PDF/A-2u conformance
- PDF/A-3a conformance
- PDF/A-3b conformance
- PDF/A-3u conformance

Note: 1. To know more details about PDF/A standard refer

<https://en.wikipedia.org/wiki/PDF/A#Description>

2. To know more details about PDF/X standard refer <https://en.wikipedia.org/wiki/PDF/X>

Note: Essential PDF supports PDF conformances only in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms.

PDF/A-1b conformance

You can create a PDF/A-1b document by specifying the conformance level as Pdf_A1B through PdfConformanceLevel Enum when creating the new PDF document, as shown below.

C#

```
//Create a new document with PDF/A-1b standard.
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A1B);
//Add a page.
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Create a solid brush.
PdfBrush brush = new PdfSolidBrush(Color.Black);
Font font = new Font("Arial", 20f, FontStyle.Regular);
//Set the font.
PdfFont pdfFont = new PdfTrueTypeFont(font, FontStyle.Regular, 12, false,
true);
//Draw the text.
graphics.DrawString("Hello world!", pdfFont, brush, new PointF(20, 20));
//Save and close the document.
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new document with PDF/A-1b standard.
Dim document As New PdfDocument(PdfConformanceLevel.Pdf_A1B)
'Add a page.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
```



```

Dim graphics As PdfGraphics = page.Graphics
'Create a solid brush.
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
Dim font As New Font("Arial", 20.0F, FontStyle.Regular)
'Set the font.
Dim pdfFont As PdfFont = New PdfTrueTypeFont(font, FontStyle.Regular, 12,
False, True)
'Draw the text.
graphics.DrawString("Hello world!", pdfFont, brush, New PointF(20, 20))
'Save and close the document.
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new document with PDF/A-1b standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A1B);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new document with PDF/A-1b standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A1B);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
FileStream fontStream = new FileStream("Arial.ttf", FileMode.Open,
FileAccess.Read);
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();

```

```
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new document with PDF/A-1b standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A1B);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

PDF/A-2b conformance

You can create a PDF/A-2b document by specifying the conformance level as Pdf_A2B through [PdfConformanceLevel](#) Enum when creating the new PDF document as follows.

C#

```

//Create a new document with PDF/A-2b standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A2B);
//Add a page
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a solid brush
PdfBrush brush = new PdfSolidBrush(Color.Black);
Font font = new Font("Arial", 20f, FontStyle.Regular);
//Set the font
PdfFont pdfFont = new PdfTrueTypeFont(font, FontStyle.Regular, 12, false,
true);
//Draw the text
graphics.DrawString("Hello world!", pdfFont, brush, new PointF(20, 20));
//Save and close the document
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a new document with PDF/A-2b standard
Dim document As New PdfDocument(PdfConformanceLevel.Pdf_A2B)
'Add a page
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create a solid brush
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
Dim font As New Font("Arial", 20.0F, FontStyle.Regular)
'Set the font
Dim pdfFont As PdfFont = New PdfTrueTypeFont(font, FontStyle.Regular, 12,
False, True)
'Draw the text
graphics.DrawString("Hello world!", pdfFont, brush, New PointF(20, 20))
'Save and close the document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new document with PDF/A-2b standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A2B);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text

```

```

graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0, 0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new document with PDF/A-2b standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A2B);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
FileStream fontStream = new FileStream("Arial.ttf", FileMode.Open,
FileAccess.Read);
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new document with PDF/A-2b standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A2B);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);

```

```

//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

PDF/A-3b conformance

The PDF/A-3b conformance supports the external files as attachment to the PDF document, so you can attach any document format such as Excel, Word, HTML, CAD, or XML files.

You can create a PDF/A-3b document by specifying the conformance level as Pdf_A3B through [PdfConformanceLevel](#) Enum when creating the new PDF document as follows.

C#

```

//Create a new document with PDF/A-3b standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
//Add a page
PdfPage page = document.Pages.Add();
//Creates an attachment
PdfAttachment attachment = new PdfAttachment("Input.txt");
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a solid brush
PdfBrush brush = new PdfSolidBrush(Color.Black);
Font font = new Font("Arial", 20f, FontStyle.Regular);
//Set the font
PdfFont pdfFont = new PdfTrueTypeFont(font, FontStyle.Regular, 12, false,
true);
//Draw the text
graphics.DrawString("Hello world!", pdfFont, brush, new PointF(20, 20));
//Save and close the document
document.Save("Output.pdf");

```

```
document.Close(true);
```

VB.NET

```
'Create a new document with PDF/A-3b standard
Dim document As New PdfDocument(PdfConformanceLevel.Pdf_A3B)
'Add a page
Dim page As PdfPage = document.Pages.Add()
'Creates an attachment
Dim attachment As New PdfAttachment("Input.txt")
attachment.Relationship = PdfAttachmentRelationship.Alternative
attachment.ModificationDate = DateTime.Now
attachment.Description = "Input.txt"
attachment.MimeType = "application/txt"
'Adds the attachment to the document
document.Attachments.Add(attachment)
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create a solid brush
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
Dim font As New Font("Arial", 20.0F, FontStyle.Regular)
'Set the font
Dim pdfFont As PdfFont = New PdfTrueTypeFont(font, FontStyle.Regular, 12,
False, True)
'Draw the text
graphics.DrawString("Hello world!", pdfFont, brush, New PointF(20, 20))
'Save and close the document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a new document with PDF/A-3b standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Creates an attachment
Stream fileStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.txt");
PdfAttachment attachment = new PdfAttachment(@"Input.txt", fileStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
```

```

graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new document with PDF/A-3b standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Creates an attachment
Stream fileStream = new FileStream("Input.txt", FileMode.Open,
FileAccess.Read);
PdfAttachment attachment = new PdfAttachment("Input.txt", fileStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
FileStream fontStream = new FileStream("Arial.ttf", FileMode.Open,
FileAccess.Read);
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new document with PDF/A-3b standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);

```

```

//Add a page to the document
PdfPage page = document.Pages.Add();
Stream fileStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.txt");
PdfAttachment attachment = new PdfAttachment("Input.txt", fileStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

PDF/A-1a conformance

PDF/A-1a conformance includes all PDF/A-1b requirements in addition to the features intended to improve a document's accessibility. PDF/A-1a conformance additionally have crucial properties of Tagged PDF.

You can create a PDF/A-1a document by specifying the conformance level as Pdf_A1A through PdfConformanceLevel Enum when creating the new PDF document, as shown below.

C#

```

//Create a new document with PDF/A-1a standard.
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A1A);
//Add a page.

```



```

PdfPage page = document.Pages.Add();
//Create PDF graphics for the page.
PdfGraphics graphics = page.Graphics;
//Create a solid brush.
PdfBrush brush = new PdfSolidBrush(Color.Black);
Font font = new Font("Arial", 20f, FontStyle.Regular);
//Set the font.
PdfFont pdfFont = new PdfTrueTypeFont(font, FontStyle.Regular, 12, false,
true);
//Draw the text.
graphics.DrawString("Hello world!", pdfFont, brush, new PointF(20, 20));
//Save and close the document.
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a new document with PDF/A-1a standard.
Dim document As New PdfDocument(PdfConformanceLevel.Pdf_A1A)
'Add a page.
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Create a solid brush.
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
Dim font As New Font("Arial", 20.0F, FontStyle.Regular)
'Set the font.
Dim pdfFont As PdfFont = New PdfTrueTypeFont(font, FontStyle.Regular, 12,
False, True)
'Draw the text.
graphics.DrawString("Hello world!", pdfFont, brush, New PointF(20, 20))
'Save and close the document.
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new document with PDF/A-1a standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A1A);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document

```

```
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new document with PDF/A-1a standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A1A);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
FileStream fontStream = new FileStream("Arial.ttf", FileMode.Open,
FileAccess.Read);
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new document with PDF/A-1a standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A1A);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
```

```
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.RuntimePlatform == Device.UWP)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
    "application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
    "application/pdf", stream);
}
```

PDF/A-2a conformance

PDF/A-2a conformance includes all PDF/A-2b requirements in addition to the features intended to improve a document's accessibility. PDF/A-2a conformance additionally have crucial properties of Tagged PDF.

You can create a PDF/A-2a document by specifying the conformance level as Pdf_A2A through [PdfConformanceLevel](#) Enum when creating the new PDF document as follows.

C#

```
//Create a new document with PDF/A-2a standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A2A);
//Add a page
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a solid brush
PdfBrush brush = new PdfSolidBrush(Color.Black);
Font font = new Font("Arial", 20f, FontStyle.Regular);
//Set the font
PdfFont pdfFont = new PdfTrueTypeFont(font, FontStyle.Regular, 12, false,
true);
//Draw the text
graphics.DrawString("Hello world!", pdfFont, brush, new PointF(20, 20));
//Save and close the document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new document with PDF/A-2a standard
Dim document As New PdfDocument(PdfConformanceLevel.Pdf_A2A)
'Add a page
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create a solid brush
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
```

```

Dim font As New Font("Arial", 20.0F, FontStyle.Regular)
'Set the font
Dim pdfFont As PdfFont = New PdfTrueTypeFont(font, FontStyle.Regular, 12,
False, True)
'Draw the text
graphics.DrawString("Hello world!", pdfFont, brush, New PointF(20, 20))
'Save and close the document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new document with PDF/A-2a standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A2A);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new document with PDF/A-2a standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A2A);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
FileStream fontStream = new FileStream("Arial.ttf", FileMode.Open,
FileAccess.Read);
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty

```

```

stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new document with PDF/A-2a standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A2A);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

PDF/A-3a conformance

PDF/A-3a conformance includes all PDF/A-3b requirements in addition to the features intended to improve a document's accessibility. PDF/A-3a conformance additionally have crucial properties of Tagged PDF.

You can create a PDF/A-3a document by specifying the conformance level as Pdf_A3A through [PdfConformanceLevel](#) Enum when creating the new PDF document as follows.

C#

```

//Create a new document with PDF/A-3a standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3A);
//Add a page
PdfPage page = document.Pages.Add();
//Creates an attachment
PdfAttachment attachment = new PdfAttachment("Input.txt");
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a solid brush
PdfBrush brush = new PdfSolidBrush(Color.Black);
Font font = new Font("Arial", 20f, FontStyle.Regular);
//Set the font
PdfFont pdfFont = new PdfTrueTypeFont(font, FontStyle.Regular, 12, false,
true);
//Draw the text
graphics.DrawString("Hello world!", pdfFont, brush, new PointF(20, 20));
//Save and close the document
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a new document with PDF/A-3a standard
Dim document As New PdfDocument(PdfConformanceLevel.Pdf_A3A)
'Add a page
Dim page As PdfPage = document.Pages.Add()
'Creates an attachment
Dim attachment As New PdfAttachment("Input.txt")
attachment.Relationship = PdfAttachmentRelationship.Alternative
attachment.ModificationDate = DateTime.Now
attachment.Description = "Input.txt"
attachment.MimeType = "application/txt"
'Adds the attachment to the document
document.Attachments.Add(attachment)
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create a solid brush
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
Dim font As New Font("Arial", 20.0F, FontStyle.Regular)
'Set the font
Dim pdfFont As PdfFont = New PdfTrueTypeFont(font, FontStyle.Regular, 12,
False, True)
'Draw the text
graphics.DrawString("Hello world!", pdfFont, brush, New PointF(20, 20))
'Save and close the document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```
//Create a new document with PDF/A-3a standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3A);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Creates an attachment
Stream fileStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.txt");
PdfAttachment attachment = new PdfAttachment(@"Input.txt", fileStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new document with PDF/A-3a standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3A);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Creates an attachment
Stream fileStream = new FileStream("Input.txt", FileMode.Open,
FileAccess.Read);
PdfAttachment attachment = new PdfAttachment("Input.txt", fileStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
```

```

FileStream fontStream = new FileStream("Arial.ttf", FileMode.Open,
    FileAccess.Read);
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
    Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
    type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new document with PDF/A-3a standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3A);
//Add a page to the document
PdfPage page = document.Pages.Add();
Stream fileStream =
    typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
    Input.txt");
PdfAttachment attachment = new PdfAttachment("Input.txt", fileStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font
Stream fontStream =
    typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
    sets.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
    Syncfusion.Drawing.PointF(0, 0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file

```



```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the PDF/Xamarin section for respective code
//samples
if (Device.RuntimePlatform == Device.UWP)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
    "application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
    "application/pdf", stream);
}
```

PDF/A-2u conformance

PDF/A-2u conformance includes all PDF/A-2b requirements, and additionally Unicode mapping for all text in the document.

You can create a PDF/A-2u document by specifying the conformance level as Pdf_A2U through [PdfConformanceLevel](#) Enum when creating the new PDF document as follows.

C#

```
//Create a new document with PDF/A-2u standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A2U);
//Add a page
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a solid brush
PdfBrush brush = new PdfSolidBrush(Color.Black);
Font font = new Font("Arial", 20f, FontStyle.Regular);
//Set the font
PdfFont pdfFont = new PdfTrueTypeFont(font, FontStyle.Regular, 12, false,
true);
//Draw the text
graphics.DrawString("Hello world!", pdfFont, brush, new PointF(20, 20));
//Save and close the document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new document with PDF/A-2u standard
Dim document As New PdfDocument(PdfConformanceLevel.Pdf_A2U)
'Add a page
Dim page As PdfPage = document.Pages.Add()
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create a solid brush
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
Dim font As New Font("Arial", 20.0F, FontStyle.Regular)
'Set the font
Dim pdfFont As PdfFont = New PdfTrueTypeFont(font, FontStyle.Regular, 12,
False, True)
```

```
'Draw the text
graphics.DrawString("Hello world!", pdfFont, brush, New PointF(20, 20))
'Save and close the document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a new document with PDF/A-2u standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A2U);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new document with PDF/A-2u standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A2U);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
FileStream fontStream = new FileStream("Arial.ttf", FileMode.Open,
FileAccess.Read);
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file
```

```
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new document with PDF/A-2u standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A2U);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

PDF/A-3u conformance

PDF/A-3u conformance includes all PDF/A-3b requirements, and additionally Unicode mapping for all text in the document.

You can create a PDF/A-3u document by specifying the conformance level as Pdf_A3U through [PdfConformanceLevel](#) Enum when creating the new PDF document as follows.

C#

```
//Create a new document with PDF/A-3u standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3U);
```

```

//Add a page
PdfPage page = document.Pages.Add();
//Creates an attachment
PdfAttachment attachment = new PdfAttachment("Input.txt");
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Create a solid brush
PdfBrush brush = new PdfSolidBrush(Color.Black);
Font font = new Font("Arial", 20f, FontStyle.Regular);
//Set the font
PdfFont pdfFont = new PdfTrueTypeFont(font, FontStyle.Regular, 12, false,
true);
//Draw the text
graphics.DrawString("Hello world!", pdfFont, brush, new PointF(20, 20));
//Save and close the document
document.Save("Output.pdf");
document.Close(true);

```

VB.NET

```

'Create a new document with PDF/A-3u standard
Dim document As New PdfDocument(PdfConformanceLevel.Pdf_A3U)
'Add a page
Dim page As PdfPage = document.Pages.Add()
'Creates an attachment
Dim attachment As New PdfAttachment("Input.txt")
attachment.Relationship = PdfAttachmentRelationship.Alternative
attachment.ModificationDate = DateTime.Now
attachment.Description = "Input.txt"
attachment.MimeType = "application/txt"
'Adds the attachment to the document
document.Attachments.Add(attachment)
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Create a solid brush
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
Dim font As New Font("Arial", 20.0F, FontStyle.Regular)
'Set the font
Dim pdfFont As PdfFont = New PdfTrueTypeFont(font, FontStyle.Regular, 12,
False, True)
'Draw the text
graphics.DrawString("Hello world!", pdfFont, brush, New PointF(20, 20))
'Save and close the document
document.Save("Output.pdf")
document.Close(True)

```

UWP

```

//Create a new document with PDF/A-3u standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3U);

```

```

//Add a page to the document
PdfPage page = document.Pages.Add();
//Creates an attachment
Stream fileStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Input.txt");
PdfAttachment attachment = new PdfAttachment(@"Input.txt", fileStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new PointF(0,
0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Create a new document with PDF/A-3u standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3U);
//Add a page to the document
PdfPage page = document.Pages.Add();
//Creates an attachment
Stream fileStream = new FileStream("Input.txt", FileMode.Open,
FileAccess.Read);
PdfAttachment attachment = new PdfAttachment("Input.txt", fileStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font from the local file
FileStream fontStream = new FileStream("Arial.ttf", FileMode.Open,
FileAccess.Read);
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text

```

```

graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a new document with PDF/A-3u standard
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3U);
//Add a page to the document
PdfPage page = document.Pages.Add();
Stream fileStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.txt");
PdfAttachment attachment = new PdfAttachment("Input.txt", fileStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "Input.txt";
attachment.MimeType = "application/txt";
//Adds the attachment to the document
document.Attachments.Add(attachment);
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the TrueType font
Stream fontStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Arial.ttf");
//Initialize the PDF TrueType font
PdfFont font = new PdfTrueTypeFont(fontStream, 14);
//Draw the text
graphics.DrawString("Hello World!!!", font, PdfBrushes.Black, new
Syncfusion.Drawing.PointF(0, 0));
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

PDF/X-1a conformance

You can create a PDF/X-1a document by specifying the conformance level as Pdf_X1A2001 through PdfConformanceLevel Enum when creating the new PDF document, as shown below.

C#

```
//Create a new document with PDF/x standard.
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_X1A2001);
//Add a page.
PdfPage page = document.Pages.Add();
document.ColorSpace = PdfColorSpace.CMYK;
//Create Pdf graphics for the page.
PdfGraphics graphics = page.Graphics;
//Create a solid brush.
PdfBrush brush = new PdfSolidBrush(Color.Black);
Font font = new Font("Arial", 20f, FontStyle.Regular);
//Set the font.
PdfFont pdfFont = new PdfTrueTypeFont(font, FontStyle.Regular, 12, false,
true);
//Draw the text.
graphics.DrawString("Hello world!", pdfFont, brush, new PointF(20, 20));
//Save and close the document.
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new document with PDF/x standard.
Dim document As New PdfDocument(PdfConformanceLevel.Pdf_X1A2001)
'Add a page.
Dim page As PdfPage = document.Pages.Add()
'set ColorSpace
document.ColorSpace = PdfColorSpace.CMYK
'Create Pdf graphics for the page.
Dim graphics As PdfGraphics = page.Graphics
'Create a solid brush.
Dim brush As PdfBrush = New PdfSolidBrush(Color.Black)
Dim font As New Font("Arial", 20.0F, FontStyle.Regular)
'Set the font.
Dim pdfFont As PdfFont = New PdfTrueTypeFont(font, FontStyle.Regular, 12,
False, True)
'Draw the text.
graphics.DrawString("Hello world!", font, brush, New PointF(20, 20))
'Save and close the document.
document.Save("Output.pdf")
document.Close(True)
```

PDF to PDF/A-1b conversion

An existing PDF document can be converted to PDF/A-1b conformance document, by setting the [Conformance](#) value in the [PdfLoadedDocument](#) to Pdf_A1B of [PdfConformanceLevel](#). Refer the below code snippet to achieve the same.

C#

```
//Load an existing PDF.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Set the conformance for PDF/A-1b conversion.
loadedDocument.Conformance = PdfConformanceLevel.Pdf_A1B;
//Save and close the document.
loadedDocument.Save("Output.pdf");
loadedDocument.Close(true);
```

VB.NET

```
'Load an existing PDF.
Dim document As New PdfLoadedDocument("Input.pdf")
'Set the conformance for PDF/A-1b conversion.
loadedDocument.Conformance = PdfConformanceLevel.Pdf_A1B
'Save and close the document.
loadedDocument.Save("Output.pdf")
loadedDocument.Close(True)
```

Note: 1. Converting PDF to PDF/X-1a conformance document is not supported.

2. CMYK color space images and symbolic fonts are not supported.

Working with ZUGFeRD invoice

The ZUGFeRD invoice is one of the uniformed data format for electronic invoices based on the ISO standard PDF/A-3, which is specifically designed for long term archiving. The ZUGFeRD invoice contains both human-readable invoice and machine-readable structured invoice data (XML).

The Essential PDF provides support to create PDF document with ZUGFeRD invoice.

Generating ZUGFeRD invoice

The Essential PDF has support to create PDF document with PDF/A-3b conformance, which allow to add external file to the PDF document as attachment.

The ZUGFeRD has two versions, ZugferdVersion 1.0 and ZugferdVersion 2.0. The Zugferd 2.0 is an updated version of Zugferd 1.0.

The ZUGFeRD has five conformance levels

- Basic: Represents the structured data for simple invoices. Additional information can be included as free text.
- Comfort: Represents the structured data for fully automated invoice processing.
- Extended: Represents the additional structured data for exchanging invoice across different industry segments.
- Minimum: Represents the basic invoice details compatible with the French Standard Factur-X.

- EN16931: Represents the fully compliant with the EU Standard, though it only defines the core elements of an invoice.

Note: * The ZUGFeRD conformance levels “Minimum” and “EN16931” are only supported in ZugferdVersion2.0.

Using PDF/A-3b conformance, you can create a ZUGFeRD invoice.

C#

```
//Create ZUGFeRD invoice PDF
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
document.ZugferdConformanceLevel = ZugferdConformanceLevel.Basic;
```

VB.NET

```
'Create ZUGFeRD invoice PDF
Dim document As PdfDocument = New PdfDocument(PdfConformanceLevel.Pdf_A3B)
document.ZugferdConformanceLevel = ZugferdConformanceLevel.Basic
```

UWP

```
//Create ZUGFeRD invoice PDF
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
document.ZugferdConformanceLevel = ZugferdConformanceLevel.Basic;
```

ASP.NET CORE

```
//Create ZUGFeRD invoice PDF
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
document.ZugferdConformanceLevel = ZugferdConformanceLevel.Basic;
```

XAMARIN

```
//Create ZUGFeRD invoice PDF
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
document.ZugferdConformanceLevel = ZugferdConformanceLevel.Basic;
```

Using PDF/A-3b conformance, you can create a ZUGFeRD invoice with ZugferdVersion2.0. By default, ZugferdVersion1.0 used.

C#

```
//Create ZUGFeRD invoice PDF
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
//Specifies ZugferdVersion
document.ZugferdVersion = ZugferdVersion.ZugferdVersion2_0;
```

VB.NET

```
'Create ZUGFeRD invoice PDF
Dim document As PdfDocument = New PdfDocument(PdfConformanceLevel.Pdf_A3B)
'Specifies ZugferdVersion
document.ZugferdVersion = ZugferdVersion.ZugferdVersion2_0
```

UWP

```
//Create ZUGFeRD invoice PDF
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
//Specifies ZugferdVersion
document.ZugferdVersion = ZugferdVersion.ZugferdVersion2_0;
```

ASP.NET CORE

```
//Create ZUGFeRD invoice PDF
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
//Specifies ZugferdVersion
document.ZugferdVersion = ZugferdVersion.ZugferdVersion2_0;
```

XAMARIN

```
//Create ZUGFeRD invoice PDF
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
//Specifies ZugferdVersion
document.ZugferdVersion = ZugferdVersion.ZugferdVersion2_0;
```

Adding ZUGFeRD structured data as attachment

The PDF/A-3 conformance supports the external files as attachment to the PDF document, so you can attach the ZUGFeRD data as attachment to the PDF document.

C#

```
//Creates an attachment
FileStream invoiceStream = new FileStream("../.. / Data / ZUGFeRD -
invoice.xml", FileMode.Open, FileAccess.Read);
PdfAttachment attachment = new PdfAttachment("ZUGFeRD-
invoice.xml", invoiceStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "ZUGFeRD-invoice";
attachment.MimeType = "application/xml";
document.Attachments.Add(attachment);
```

VB.NET

```
'Creates an attachment
Dim invoiceStream As FileStream = New FileStream("../.. / Data / ZUGFeRD -
invoice.xml", FileMode.Open, FileAccess.Read)
Dim attachment As PdfAttachment = New PdfAttachment("ZUGFeRD-invoice.xml",
invoiceStream)
attachment.Relationship = PdfAttachmentRelationship.Alternative
attachment.ModificationDate = DateTime.Now
attachment.Description = "ZUGFeRD-invoice"
attachment.MimeType = "application/xml"
document.Attachments.Add(attachment)
```

UWP

```
//Creates an attachment
Stream invoiceStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.ZUGFeRD-invoice.xml");
PdfAttachment attachment = new PdfAttachment("ZUGFeRD-
invoice.xml", invoiceStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "ZUGFeRD-invoice";
attachment.MimeType = "application/xml";
document.Attachments.Add(attachment);
```

ASP.NET CORE

```
//Creates an attachment
FileStream fontStream = new FileStream("../.. / Data / ZUGFeRD -
invoice.xml", FileMode.Open, FileAccess.Read);
PdfAttachment attachment = new PdfAttachment("ZUGFeRD-
invoice.xml", invoiceStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "ZUGFeRD-invoice";
attachment.MimeType = "application/xml";
document.Attachments.Add(attachment);
```

XAMARIN

```
//Creates an attachment
Stream invoiceStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.ZUGFeRD-invoice.xml");
PdfAttachment attachment = new PdfAttachment("ZUGFeRD-
invoice.xml", invoiceStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "ZUGFeRD-invoice";
attachment.MimeType = "application/xml";
document.Attachments.Add(attachment);
```

Note: * As per the ZUGFeRD standard guidelines, the XML name must be ZUGFeRD-invoice.xml.

Complete code

The complete code snippet is given as follows.

C#

```
//Create ZUGFeRD invoice PDF
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
document.ZugferdConformanceLevel = ZugferdConformanceLevel.Basic;
//Creates an attachment
FileStream invoiceStream = new FileStream("../.. / Data / ZUGFeRD -
invoice.xml", FileMode.Open, FileAccess.Read);
PdfAttachment attachment = new PdfAttachment("ZUGFeRD-
invoice.xml", invoiceStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
```

```
attachment.ModificationDate = DateTime.Now;
attachment.Description = "ZUGFeRD-invoice";
attachment.MimeType = "application/xml";
document.Attachments.Add(attachment);
// Save and close the document
document.Save("Zugferd.pdf");
document.Close(true);
```

VB.NET

```
'Create ZUGFeRD invoice PDF
Dim document As PdfDocument = New PdfDocument(PdfConformanceLevel.Pdf_A3B)
document.ZugferdConformanceLevel = ZugferdConformanceLevel.Basic
'Creates an attachment
Dim invoiceStream As FileStream = New FileStream("../.. / Data / ZUGFeRD -
invoice.xml", FileMode.Open, FileAccess.Read)
Dim attachment As PdfAttachment = New PdfAttachment("ZUGFeRD-
invoice.xml", invoiceStream)
attachment.Relationship = PdfAttachmentRelationship.Alternative
attachment.ModificationDate = DateTime.Now
attachment.Description = "ZUGFeRD-invoice"
attachment.MimeType = "application/xml"
document.Attachments.Add(attachment)
'Save and close the document
document.Save("Zugferd.pdf")
document.Close(True)
```

UWP

```
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
document.ZugferdConformanceLevel = ZugferdConformanceLevel.Basic;
//Creates an attachment
Stream invoiceStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.ZUGFeRD-invoice.xml");
PdfAttachment attachment = new PdfAttachment("ZUGFeRD-
invoice.xml", invoiceStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "ZUGFeRD-invoice";
attachment.MimeType = "application/xml";
document.Attachments.Add(attachment);
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respected code samples
Save(stream, "Zugferd.pdf");
```

ASP.NET CORE

```
PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
document.ZugferdConformanceLevel = ZugferdConformanceLevel.Basic;
//Creates an attachment
FileStream fontStream = new FileStream("../.. / Data / ZUGFeRD -
invoice.xml", FileMode.Open, FileAccess.Read);
```

```

PdfAttachment attachment = new PdfAttachment("ZUGFeRD-
invoice.xml",invoiceStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "ZUGFeRD-invoice";
attachment.MimeType = "application/xml";
document.Attachments.Add(attachment);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document into memory stream
document.Save(stream);
//If the position is not set to '0', then the PDF will be empty
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Zugferd.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

PdfDocument document = new PdfDocument(PdfConformanceLevel.Pdf_A3B);
document.ZugferdConformanceLevel = ZugferdConformanceLevel.Basic;
//Creates an attachment
Stream invoiceStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.ZUGFeRD-invoice.xml");
PdfAttachment attachment = new PdfAttachment("ZUGFeRD-
invoice.xml",invoiceStream);
attachment.Relationship = PdfAttachmentRelationship.Alternative;
attachment.ModificationDate = DateTime.Now;
attachment.Description = "ZUGFeRD-invoice";
attachment.MimeType = "application/xml";
document.Attachments.Add(attachment);
//Save the document into memory stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Zugferd.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Zugferd.pdf",
"application/pdf", stream);
}

```

```
}

```

Extract ZUGFeRD invoice from PDF

You can extract the ZUGFeRD invoice using the following code.

C#

```
//Loads the PDF document
PdfLoadedDocument document = new PdfLoadedDocument("Sample.pdf");
//Iterates the attachments
foreach (PdfAttachment attachment in document.Attachments)
{
    //Extracts the ZUGFeRD invoice attachment and saves it to the disk
    FileStream s = new FileStream(attachment.FileName, FileMode.Create);
    s.Write(attachment.Data, 0, attachment.Data.Length);
    s.Dispose();
}
//Saves and closes the document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Loads the PDF document
Dim document As New PdfLoadedDocument("Sample.pdf")
'Iterates the attachments
For Each attachment As PdfAttachment In document.Attachments
    'Extracts the attachment and saves it to the disk
    Dim s As New FileStream(attachment.FileName, FileMode.Create)
    s.Write(attachment.Data, 0, attachment.Data.Length)
    s.Dispose()
Next
'Saves and closes the document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Iterates the attachments
foreach (PdfAttachment attachment in loadedDocument.Attachments)
{
    //Extracts the ZUGFeRD invoice attachment and saves it to the disk
    FileStream s = new FileStream(attachment.FileName, FileMode.Create);
    s.Write(attachment.Data, 0, attachment.Data.Length);
    s.Dispose();
}
```

```

}
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Iterates the attachments
foreach (PdfAttachment attachment in loadedDocument.Attachments)
{
//Extracts the ZUGFeRD invoice attachment and saves it to the disk
FileStream s = new FileStream(attachment.FileName, FileMode.Create);
s.Write(attachment.Data, 0, attachment.Data.Length);
s.Dispose();
}
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Closes the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Iterates the attachments
foreach (PdfAttachment attachment in loadedDocument.Attachments)
{
//Extracts the ZUGFeRD invoice attachment and saves it to the disk
FileStream s = new FileStream(attachment.FileName, FileMode.Create);
s.Write(attachment.Data, 0, attachment.Data.Length);
s.Dispose();
}
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);

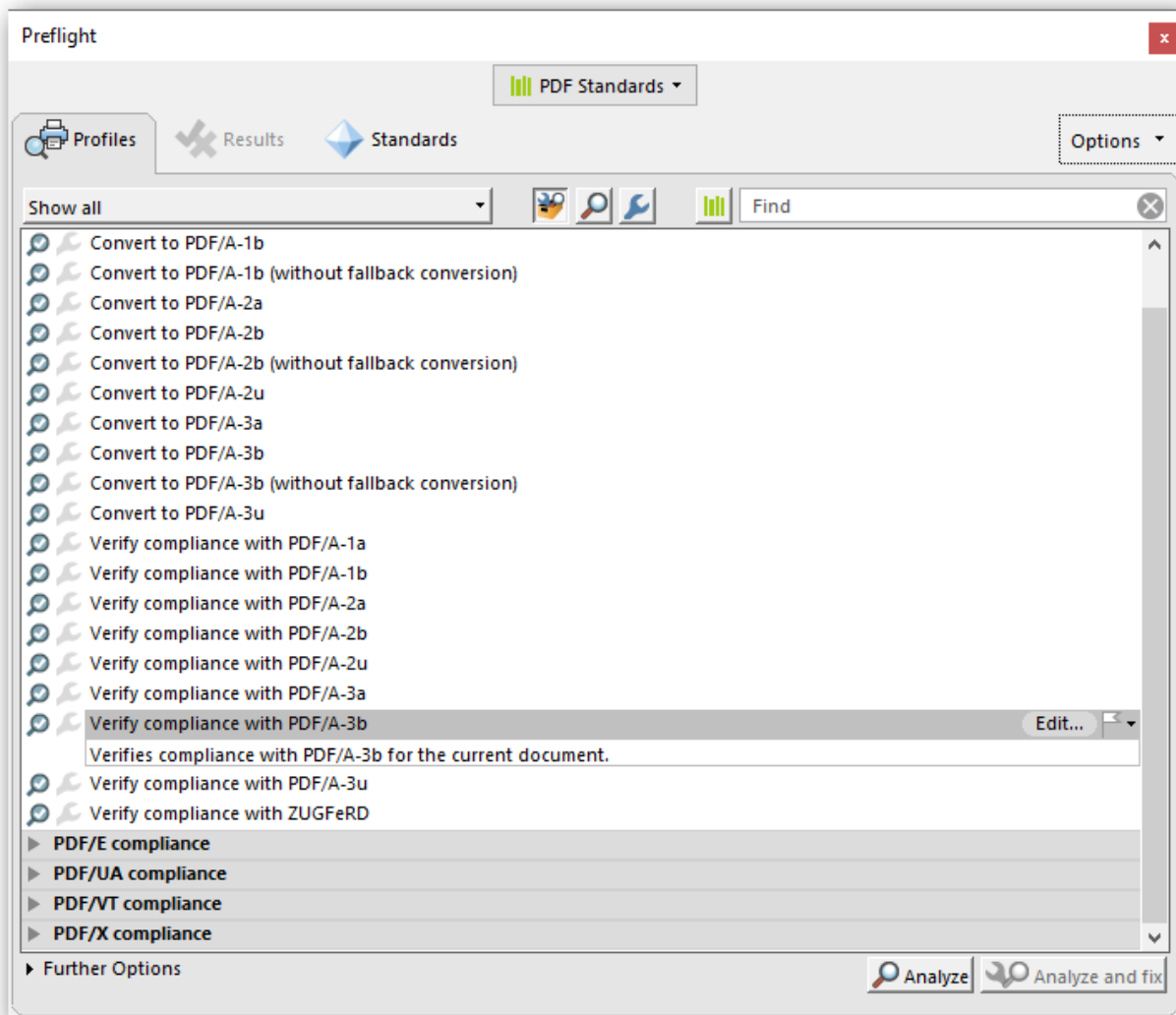
```

```
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("output.pdf",
"application/pdf", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("output.pdf",
"application/pdf", stream);
}
```

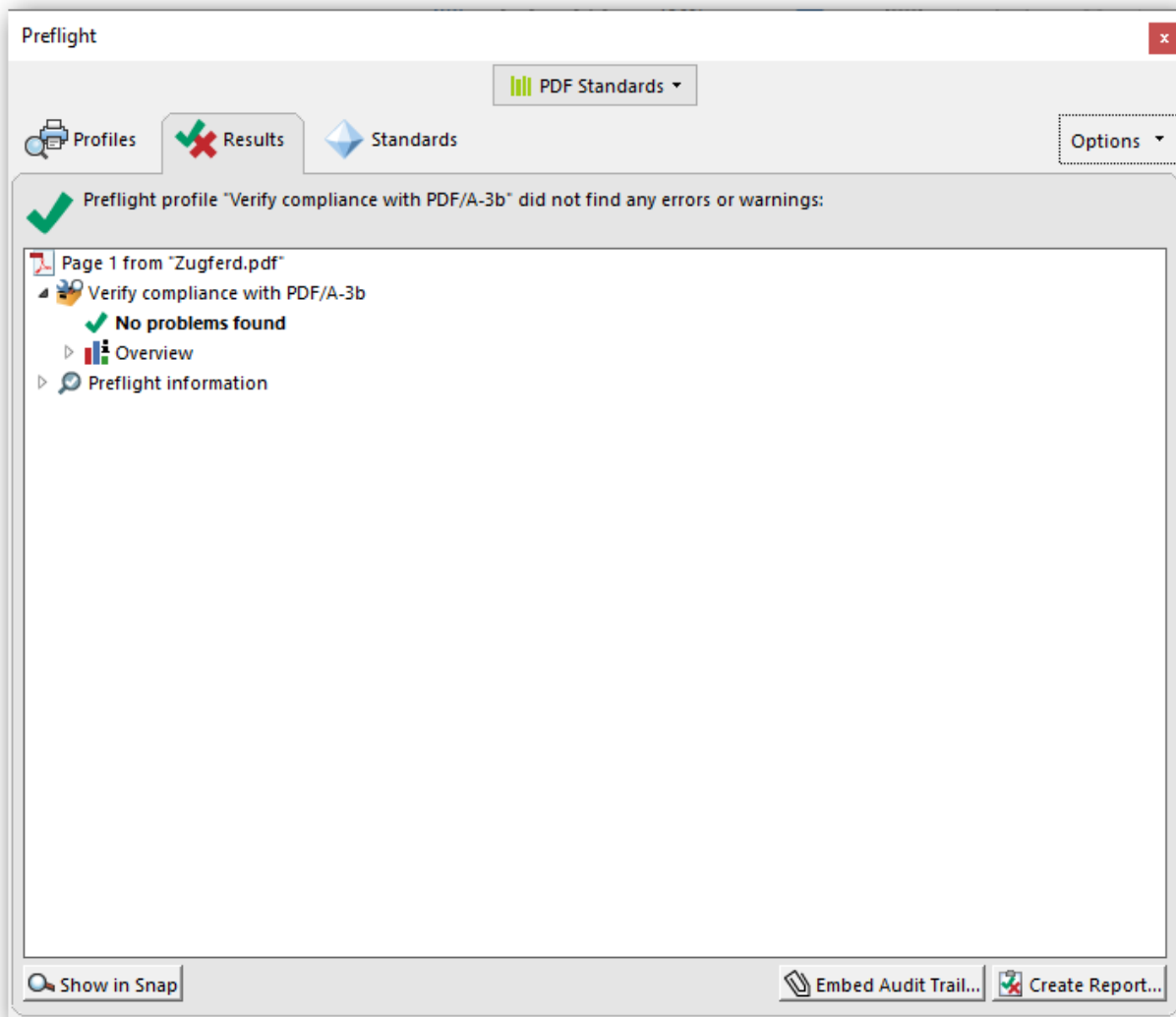
Validating ZUGFeRD invoices using Adobe Acrobat

The ZUGFeRD invoices can be validated as follows:

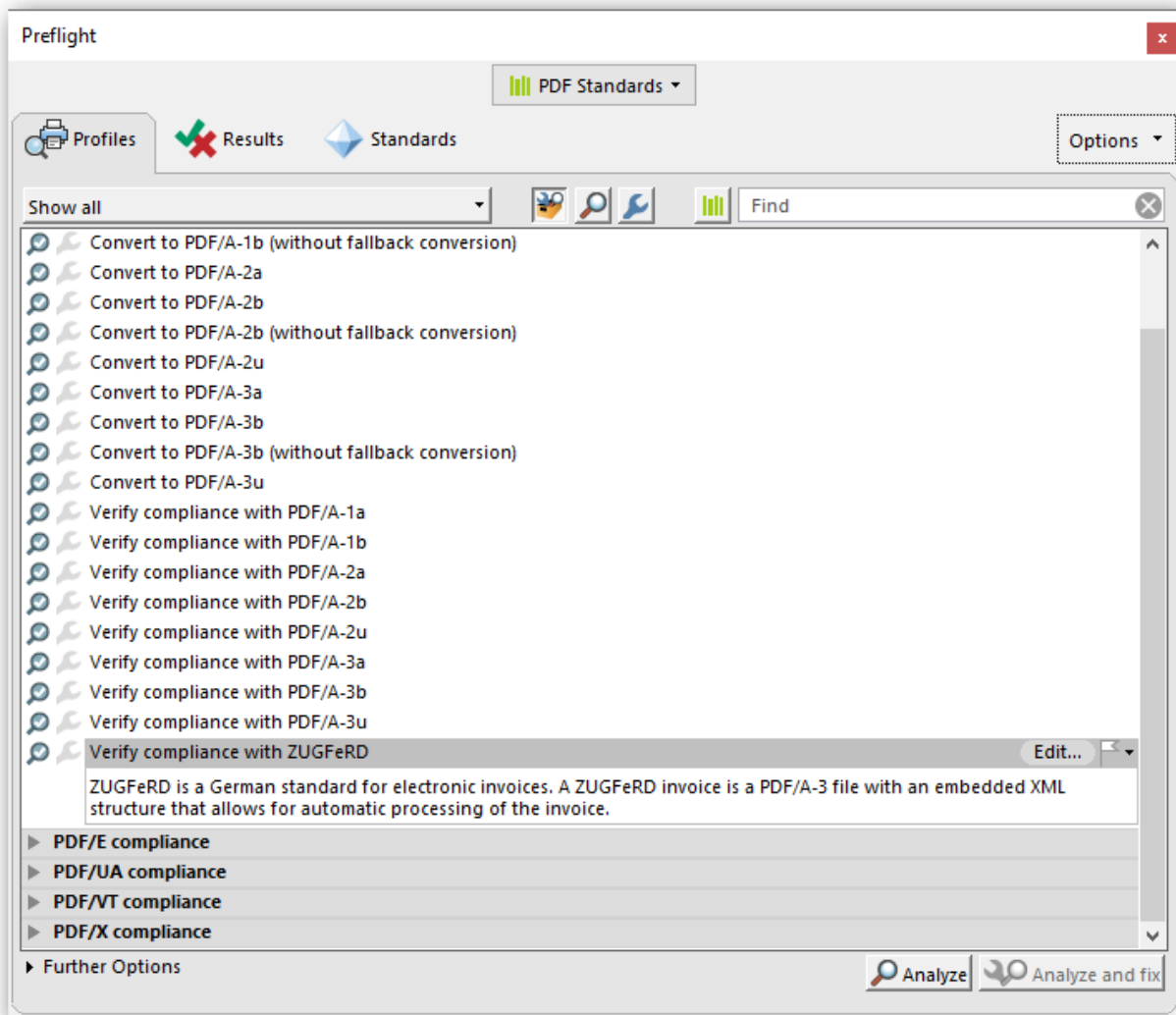
1. Conformance to the PDF/A-3b standard can be checked with Preflight function in Adobe acrobat.
 - Open the PDF and choose Tools > Print Production > Preflight in the right pane.
 - Select the PDF/A-3b profile and click the Analyze button.



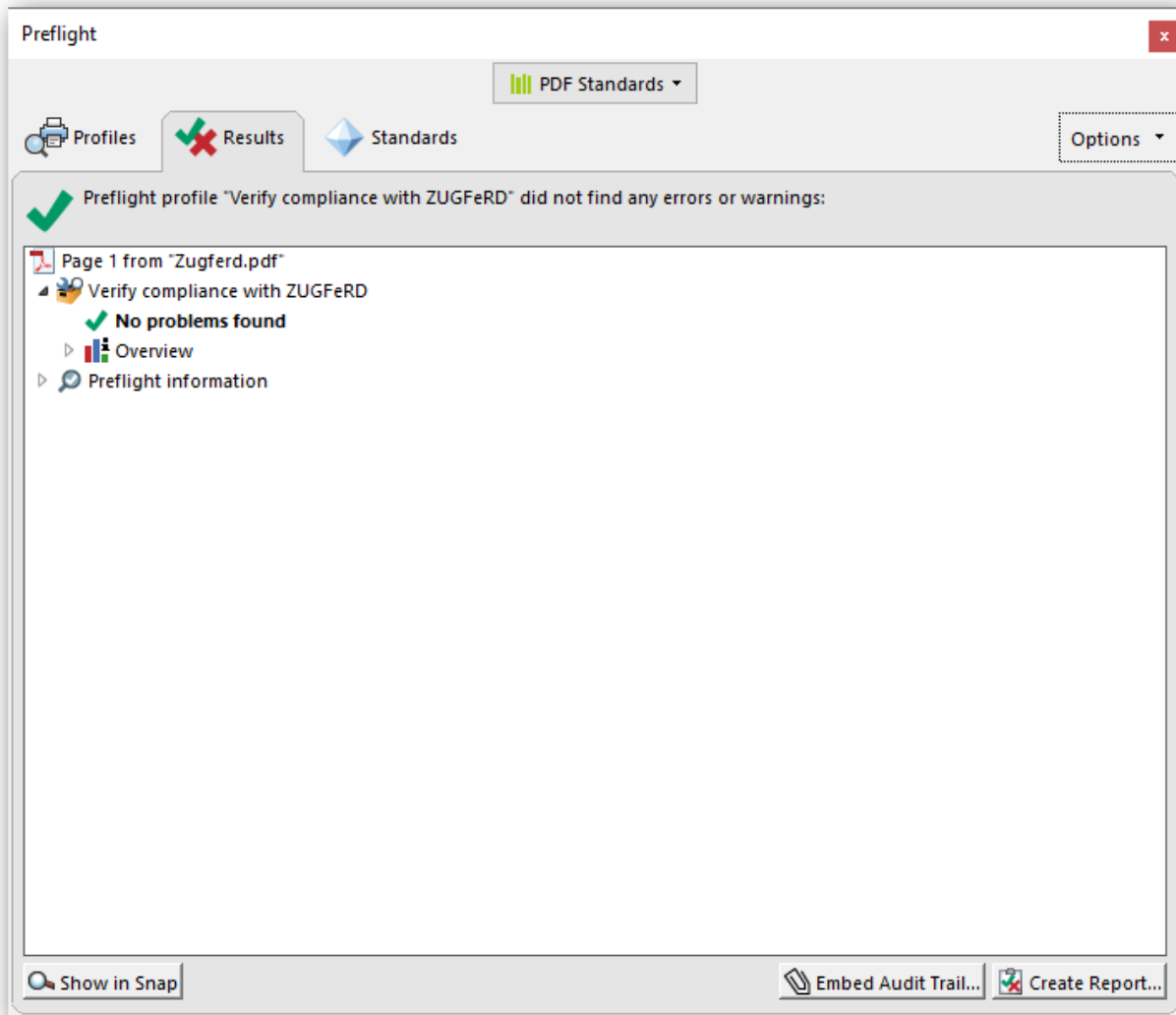
- Check the result



2. Conformance to the ZUGFeRD invoice can be checked as follows: *Open the PDF and choose Tools > Print Production > Preflight in the right pane.* Select the ZUGFeRD profile and click the Analyze button.



- Check the result



Working with Metadata (XMP)

Metadata is a data that describes the characteristics or properties of a document.

Metadata includes document information properties such as author, modification date, and copyright status.

Working with the XMP metadata

In order to work multiple applications effectively with metadata, there must be a common standard that they understand. XMP-the Extensible Metadata Platform is designed to provide such a standard.

XMP standardizes the definition, creation, and processing of metadata.

Adding XMP metadata in a PDF document

You can add XMP metadata in a PDF document using [XmpMetadata](#) class as shown in the code snippet below.

C#

```
//Create a PDF document  
PdfDocument pdfDoc = new PdfDocument();
```

```

//Create a page
PdfPage page = pdfDoc.Pages.Add();
//Get XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Basic Schema
BasicSchema basic = metaData.BasicSchema;
//set the basic details of the document
basic.Advisory.Add("advisory");
basic.BaseURL = new Uri("http://google.com");
basic.CreateDate = DateTime.Now;
basic.CreatorTool = "creator tool";
basic.Identifier.Add("identifier");
basic.Label = "label";
basic.MetadataDate = DateTime.Now;
basic.ModifyDate = DateTime.Now;
basic.Nickname = "nickname";
basic.Rating.Add(-25);
//save the document
pdfDoc.Save("DocumentInformation.pdf");
pdfDoc.Close(true);

```

VB.NET

```

'Create a PDF document
Dim pdfDoc As New PdfDocument()
'Create a page
Dim page As PdfPage = pdfDoc.Pages.Add()
'Get XMP object
Dim metaData As XmpMetadata = pdfDoc.DocumentInformation.XmpMetadata
'XMP Basic Schema
Dim basic As BasicSchema = metaData.BasicSchema
'set the basic details of the document
basic.Advisory.Add("advisory")
basic.BaseURL = New Uri("http://google.com")
basic.CreateDate = DateTime.Now
basic.CreatorTool = "creator tool"
basic.Identifier.Add("identifier")
basic.Label = "label"
basic.MetadataDate = DateTime.Now
basic.ModifyDate = DateTime.Now
basic.Nickname = "nickname"
basic.Rating.Add(-25)
'save the document
pdfDoc.Save("DocumentInformation.pdf")
pdfDoc.Close(True)

```

UWP

```

//Essential PDF supports Metadata (XMP) only in Windows Forms, WPF, ASP.NET, ASP.NET MVC, ASP.NET Core and Xamarin platforms

```

ASP.NET CORE

```

//Create a PDF document
PdfDocument pdfDoc = new PdfDocument();

```

```
//Create a page
PdfPage page = pdfDoc.Pages.Add();
//Get XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Basic Schema
BasicSchema basic = metaData.BasicSchema;
//Set the basic details of the document
basic.Advisory.Add("advisory");
basic.BaseURL = new Uri("http://google.com");
basic.CreateDate = DateTime.Now;
basic.CreatorTool = "creator tool";
basic.Identifier.Add("identifier");
basic.Label = "label";
basic.MetadataDate = DateTime.Now;
basic.ModifyDate = DateTime.Now;
basic.Nickname = "nickname";
basic.Rating.Add(-25);
//Save and close the document
MemoryStream stream = new MemoryStream();
pdfDoc.Save(stream);
stream.Position = 0;
//Close the document
pdfDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "DocumentInformation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a PDF document
PdfDocument pdfDoc = new PdfDocument();
//Create a page
PdfPage page = pdfDoc.Pages.Add();
//Get XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Basic Schema
BasicSchema basic = metaData.BasicSchema;
//Set the basic details of the document
basic.Advisory.Add("advisory");
basic.BaseURL = new Uri("http://google.com");
basic.CreateDate = DateTime.Now;
basic.CreatorTool = "creator tool";
basic.Identifier.Add("identifier");
basic.Label = "label";
basic.MetadataDate = DateTime.Now;
basic.ModifyDate = DateTime.Now;
basic.Nickname = "nickname";
basic.Rating.Add(-25);
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
pdfDoc.Save(memoryStream);
//Close the documents
```

```
pdfDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DocumentInfor
mation.pdf", "application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("DocumentInformation.pdf",
"application/pdf", memoryStream);
}
```

Adding XMP metadata in an existing PDF document

You can add metadata in an existing PDF document using [XmpMetadata](#) class, as follows.

C#

```
//Load the document
PdfLoadedDocument pdfDoc = new PdfLoadedDocument("input.pdf");
// Get XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
// XMP Basic Schema
BasicSchema basic = metaData.BasicSchema;
//set the basic details of the document
basic.Advisory.Add("advisory");
basic.BaseURL = new Uri("http://google.com");
basic.CreateDate = DateTime.Now;
basic.CreatorTool = "creator tool";
basic.Identifier.Add("identifier");
basic.Label = "label";
basic.MetadataDate = DateTime.Now;
basic.ModifyDate = DateTime.Now;
basic.Nickname = "nickname";
basic.Rating.Add(-25);
//save the document
pdfDoc.Save("DocumentInformation.pdf");
pdfDoc.Close(true);
```

VB.NET

```
'Load the document
Dim pdfDoc As New PdfLoadedDocument("input.pdf")
'Get metaData object
Dim metaData As XmpMetadata = pdfDoc.DocumentInformation.XmpMetadata
'XMP Basic Schema
Dim basic As BasicSchema = metaData.BasicSchema
'Set the basic details of the document
basic.Advisory.Add("advisory")
basic.BaseURL = New Uri("http://google.com")
basic.CreateDate = DateTime.Now
basic.CreatorTool = "creator tool"
```

```

basic.Identifier.Add("identifier")
basic.Label = "label"
basic.MetadataDate = DateTime.Now
basic.ModifyDate = DateTime.Now
basic.Nickname = "nickname"
basic.Rating.Add(-25)
'Save the document
pdfDoc.Save("DocumentInformation.pdf")
pdfDoc.Close(True)

```

UWP

```

//Essential PDF supports Metadata (XMP) only in Windows Forms, WPF, ASP.NET,
ASP.NET MVC, ASP.NET Core and Xamarin platforms

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument pdfDoc = new PdfLoadedDocument(docStream);
//Get XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Basic Schema
BasicSchema basic = metaData.BasicSchema;
//Set the basic details of the document
basic.Advisory.Add("advisory");
basic.BaseURL = new Uri("http://google.com");
basic.CreateDate = DateTime.Now;
basic.CreatorTool = "creator tool";
basic.Identifier.Add("identifier");
basic.Label = "label";
basic.MetadataDate = DateTime.Now;
basic.ModifyDate = DateTime.Now;
basic.Nickname = "nickname";
basic.Rating.Add(-25);
//Save and close the document
MemoryStream stream = new MemoryStream();
pdfDoc.Save(stream);
stream.Position = 0;
//Close the document.
pdfDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "DocumentInformation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream

```



```

Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument pdfDoc = new PdfLoadedDocument(docStream);
//Get XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Basic Schema
BasicSchema basic = metaData.BasicSchema;
//Set the basic details of the document
basic.Advisory.Add("advisory");
basic.BaseURL = new Uri("http://google.com");
basic.CreateDate = DateTime.Now;
basic.CreatorTool = "creator tool";
basic.Identifier.Add("identifier");
basic.Label = "label";
basic.MetadataDate = DateTime.Now;
basic.ModifyDate = DateTime.Now;
basic.Nickname = "nickname";
basic.Rating.Add(-25);
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
pdfDoc.Save(memoryStream);
//Close the documents
pdfDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DocumentInfor
mation.pdf", "application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("DocumentInformation.pdf",
"application/pdf", memoryStream);
}

```

Supported Schema types

XMP is provided with the following schemas:

- Basic Schema
- Dublin Core Schema
- Rights Management Schema
- Basic Job Ticket Schema
- Paged-Text Schema
- PDF Schema

Basic Schema

Basic Schema contains properties that provide basic descriptive information such as,

- Base URL
- Creation date
- Creator tool
- Label
- Modified date.
- Meta data date
- Nickname
- Rating

[BasicSchema](#) class is used to create the basic schema properties.

Refer the following code sample to create XMP basic schema.

C#

```
//Create a PDF document
PdfDocument pdfDoc = new PdfDocument();
//Create a page
PdfPage page = pdfDoc.Pages.Add();
//Get metaData object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Basic Schema
BasicSchema basic = metaData.BasicSchema;
//Set the basic details of the document
basic.Advisory.Add("advisory");
basic.BaseURL = new Uri("http://google.com");
basic.CreateDate = DateTime.Now;
basic.CreatorTool = "creator tool";
basic.Identifier.Add("identifier");
basic.Label = "label";
basic.MetadataDate = DateTime.Now;
basic.ModifyDate = DateTime.Now;
basic.Nickname = "nickname";
basic.Rating.Add(-25);
//Save the document
pdfDoc.Save("DocumentInformation.pdf");
pdfDoc.Close(true);
```

VB.NET

```
'Create a PDF document
Dim pdfDoc As New PdfDocument()
'Create a page
Dim page As PdfPage = pdfDoc.Pages.Add()
'Get metaData object
Dim metaData As XmpMetadata = pdfDoc.DocumentInformation.XmpMetadata
'XMP Basic Schema
Dim basic As BasicSchema = metaData.BasicSchema
'Set the basic details of the document
basic.Advisory.Add("advisory")
basic.BaseURL = New Uri("http://google.com")
basic.CreateDate = DateTime.Now
basic.CreatorTool = "creator tool"
basic.Identifier.Add("identifier")
basic.Label = "label"
```

```

basic.MetadataDate = DateTime.Now
basic.ModifyDate = DateTime.Now
basic.Nickname = "nickname"
basic.Rating.Add(-25)
'Save the document
pdfDoc.Save("DocumentInformation.pdf")
pdfDoc.Close(True)

```

UWP

//Essential PDF supports Metadata (XMP) only in Windows Forms, WPF, ASP.NET, ASP.NET MVC, ASP.NET Core and Xamarin platforms

ASP.NET CORE

```

//Create a PDF document
PdfDocument pdfDoc = new PdfDocument();
//Create a page
PdfPage page = pdfDoc.Pages.Add();
//Get metaData object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Basic Schema
BasicSchema basic = metaData.BasicSchema;
//Set the basic details of the document
basic.Advisory.Add("advisory");
basic.BaseURL = new Uri("http://google.com");
basic.CreateDate = DateTime.Now;
basic.CreatorTool = "creator tool";
basic.Identifier.Add("identifier");
basic.Label = "label";
basic.MetadataDate = DateTime.Now;
basic.ModifyDate = DateTime.Now;
basic.Nickname = "nickname";
basic.Rating.Add(-25);
//Save and close the document
MemoryStream stream = new MemoryStream();
pdfDoc.Save(stream);
stream.Position = 0;
//Close the document
pdfDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "DocumentInformation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a PDF document
PdfDocument pdfDoc = new PdfDocument();
//Create a page
PdfPage page = pdfDoc.Pages.Add();
//Get metaData object

```

```

XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Basic Schema
BasicSchema basic = metaData.BasicSchema;
//Set the basic details of the document
basic.Advisory.Add("advisory");
basic.BaseURL = new Uri("http://google.com");
basic.CreateDate = DateTime.Now;
basic.CreatorTool = "creator tool";
basic.Identifier.Add("identifier");
basic.Label = "label";
basic.MetadataDate = DateTime.Now;
basic.ModifyDate = DateTime.Now;
basic.Nickname = "nickname";
basic.Rating.Add(-25);
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
pdfDoc.Save(memoryStream);
//Close the documents
pdfDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DocumentInfor
mation.pdf", "application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("DocumentInformation.pdf",
"application/pdf", memoryStream);
}

```

Dublin Core Schema

The Dublin Core schema provides a set of commonly used properties such as,

- Contributor
- Coverage
- Creator
- Date
- Description
- Format
- Language
- Publisher
- Title

[DublinCoreSchema](#) class is used to create the Dublin core schema properties.

C#

```

//Create new PDF document
PdfDocument pdfDoc = new PdfDocument();

```

```

PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Dublin core Schema
DublinCoreSchema dublin = metaData.DublinCoreSchema;
//Set the Dublin Core Schema details of the document
dublin.Creator.Add("Syncfusion");
dublin.Description.Add("Title", "Essential PDF creator");
dublin.Title.Add("Resource name", "Documentation");
dublin.Type.Add("PDF");
dublin.Publisher.Add("Essential PDF");
//Saves the document
pdfDoc.Save("DocumentInformation.pdf");
pdfDoc.Close(true);

```

VB.NET

```

'Create new PDF document
Dim pdfDoc As New PdfDocument()
Dim page As PdfPage = pdfDoc.Pages.Add()
'Gets XMP object
Dim metaData As XmpMetadata = pdfDoc.DocumentInformation.XmpMetadata
'XMP Dublin core Schema
Dim dublin As DublinCoreSchema = metaData.DublinCoreSchema
'Set the Dublin Core Schema details of the document
dublin.Creator.Add("Syncfusion")
dublin.Description.Add("Title", "Essential PDF creator")
dublin.Title.Add("Resource name", "Documentation")
dublin.Type.Add("PDF")
dublin.Publisher.Add("Essential PDF")
'Saves the document
pdfDoc.Save("DocumentInformation.pdf")
pdfDoc.Close(True)

```

UWP

```

//Essential PDF supports Metadata (XMP) only in Windows Forms, WPF, ASP.NET, ASP.NET MVC, ASP.NET Core and Xamarin platforms

```

ASP.NET CORE

```

//Create new PDF document
PdfDocument pdfDoc = new PdfDocument();
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Dublin core Schema
DublinCoreSchema dublin = metaData.DublinCoreSchema;
//Set the Dublin Core Schema details of the document
dublin.Creator.Add("Syncfusion");
dublin.Description.Add("Title", "Essential PDF creator");
dublin.Title.Add("Resource name", "Documentation");
dublin.Type.Add("PDF");
dublin.Publisher.Add("Essential PDF");
//Save and close the document

```

```

MemoryStream stream = new MemoryStream();
pdfDoc.Save(stream);
stream.Position = 0;
//Close the document
pdfDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "DocumentInformation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create new PDF document
PdfDocument pdfDoc = new PdfDocument();
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Dublin core Schema
DublinCoreSchema dublin = metaData.DublinCoreSchema;
//Set the Dublin Core Schema details of the document
dublin.Creator.Add("Syncfusion");
dublin.Description.Add("Title", "Essential PDF creator");
dublin.Title.Add("Resource name", "Documentation");
dublin.Type.Add("PDF");
dublin.Publisher.Add("Essential PDF");
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
pdfDoc.Save(memoryStream);
//Close the documents
pdfDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DocumentInfor
mation.pdf", "application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("DocumentInformation.pdf",
"application/pdf", memoryStream);
}

```

Rights Management Schema

This schema includes properties related to rights management. These properties provide information regarding the legal restrictions associated with a resource.

- Certificate

- Marked
- Owner
- UsageTerm
- WebStatement

[RightsManagementSchema](#) class is used to create the Rights management schema properties.

C#

```
//Create PDF document
PdfDocument pdfDoc = new PdfDocument();
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Rights Management Schema
RightsManagementSchema rights = metaData.RightsManagementSchema;
//Set the Rights Management Schema details of the document
rights.Certificate = new Uri("http://syncfusion.com");
rights.Owner.Add("Syncfusion");
rights.Marked = true;
//Save and close the document
pdfDoc.Save("DocumentInformation.pdf");
pdfDoc.Close(true);
```

VB.NET

```
'Create PDF document
Dim pdfDoc As New PdfDocument()
Dim page As PdfPage = pdfDoc.Pages.Add()
'Gets XMP object
Dim metaData As XmpMetadata = pdfDoc.DocumentInformation.XmpMetadata
'XMP Rights Management Schema
Dim rights As RightsManagementSchema = metaData.RightsManagementSchema
'Set the Rights Management Schema details of the document
rights.Certificate = New Uri("http://syncfusion.com")
rights.Owner.Add("Syncfusion")
rights.Marked = True
'Save and close the document
pdfDoc.Save("DocumentInformation.pdf")
pdfDoc.Close(True)
```

UWP

```
//Essential PDF does not support Metadata (XMP) in UWP platform
```

ASP.NET CORE

```
//Create PDF document
PdfDocument pdfDoc = new PdfDocument();
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Rights Management Schema
RightsManagementSchema rights = metaData.RightsManagementSchema;
//Set the Rights Management Schema details of the document
```

```

rights.Certificate = new Uri("http://syncfusion.com");
rights.Owner.Add("Syncfusion");
rights.Marked = true;
//Save and close the document
MemoryStream stream = new MemoryStream();
pdfDoc.Save(stream);
stream.Position = 0;
//Close the document
pdfDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "DocumentInformation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create PDF document
PdfDocument pdfDoc = new PdfDocument();
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Rights Management Schema
RightsManagementSchema rights = metaData.RightsManagementSchema;
//Set the Rights Management Schema details of the document
rights.Certificate = new Uri("http://syncfusion.com");
rights.Owner.Add("Syncfusion");
rights.Marked = true;
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
pdfDoc.Save(memoryStream);
//Close the documents
pdfDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DocumentInfor
mation.pdf", "application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("DocumentInformation.pdf",
"application/pdf", memoryStream);
}

```

Basic Job Ticket Schema

This schema describes very simple workflow or job information and [BasicJobTicketSchema](#) class is used for creation.

- JobRef

C#

```
//Create a document
PdfDocument pdfDoc = new PdfDocument();
//Add a page
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Rights Management Schema
BasicJobTicketSchema basicJob = metaData.BasicJobTicketSchema;
//Set the Rights Management Schema details of the document
basicJob.JobRef.Add("PDF document creation");
//Save the document
pdfDoc.Save("DocumentInformation.pdf");
pdfDoc.Close(true);
```

VB.NET

```
'Create a document
Dim pdfDoc As New PdfDocument()
'Add a page
Dim page As PdfPage = pdfDoc.Pages.Add()
'Gets XMP object
Dim metaData As XmpMetadata = pdfDoc.DocumentInformation.XmpMetadata
'XMP Rights Management Schema
Dim basicJob As BasicJobTicketSchema = metaData.BasicJobTicketSchema
'Set the Rights Management Schema details of the document
basicJob.JobRef.Add("PDF document creation")
'Save the document
pdfDoc.Save("DocumentInformation.pdf")
pdfDoc.Close(True)
```

UWP

```
//Essential PDF supports Metadata (XMP) only in Windows Forms, WPF, ASP.NET,
ASP.NET MVC, ASP.NET Core and Xamarin platforms
```

ASP.NET CORE

```
//Create a document
PdfDocument pdfDoc = new PdfDocument();
//Add a page
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Rights Management Schema
BasicJobTicketSchema basicJob = metaData.BasicJobTicketSchema;
//Set the Rights Management Schema details of the document
basicJob.JobRef.Add("PDF document creation");
//Save and close the document
MemoryStream stream = new MemoryStream();
pdfDoc.Save(stream);
stream.Position = 0;
```

```
//Close the document.
pdfDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "DocumentInformation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a document
PdfDocument pdfDoc = new PdfDocument();
//Add a page
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Rights Management Schema
BasicJobTicketSchema basicJob = metaData.BasicJobTicketSchema;
//Set the Rights Management Schema details of the document
basicJob.JobRef.Add("PDF document creation");
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
pdfDoc.Save(memoryStream);
//Close the documents
pdfDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DocumentInfor
mation.pdf", "application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("DocumentInformation.pdf",
"application/pdf", memoryStream);
}
```

Paged-Text Schema

The Paged-Text schema is used for text appearance on page in a document.

- MaxPageSize
- NPages
- Colorants
- PlateNames

[PagedTextSchema](#) class is used for creating Paged-Text schema properties.

C#

```

//Create a Pdf document
PdfDocument pdfDoc = new PdfDocument();
//Create a Page
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Page text Schema
PagedTextSchema pagedText = metaData.PagedTextSchema;
//Sets the Page text Schema details of the document
pagedText.MaxPageSize.Width = 500;
pagedText.MaxPageSize.Height = 750;
pagedText.NPages = 1;
pagedText.PlateNames.Add("Sample page");
//Saves the document
pdfDoc.Save("DocumentInformation.pdf");
pdfDoc.Close(true);

```

VB.NET

```

'Create a Pdf document
Dim pdfDoc As New PdfDocument()
'Create a Page
Dim page As PdfPage = pdfDoc.Pages.Add()
'Gets XMP object
Dim metaData As XmpMetadata = pdfDoc.DocumentInformation.XmpMetadata
'XMP Page text Schema
Dim pagedText As PagedTextSchema = metaData.PagedTextSchema
'Sets the Page text Schema details of the document
pagedText.MaxPageSize.Width = 500
pagedText.MaxPageSize.Height = 750
pagedText.NPages = 1
pagedText.PlateNames.Add("Sample page")
'Saves the document
pdfDoc.Save("DocumentInformation.pdf")
pdfDoc.Close(True)

```

UWP

```

//Essential PDF supports Metadata (XMP) only in Windows Forms, WPF, ASP.NET, ASP.NET MVC, ASP.NET Core and Xamarin platforms

```

ASP.NET CORE

```

//Create a Pdf document
PdfDocument pdfDoc = new PdfDocument();
//Create a Page
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Page text Schema
PagedTextSchema pagedText = metaData.PagedTextSchema;
//Sets the Page text Schema details of the document
pagedText.MaxPageSize.Width = 500;
pagedText.MaxPageSize.Height = 750;
pagedText.NPages = 1;

```

```

pagedText.PlateNames.Add("Sample page");
//Save and close the document
MemoryStream stream = new MemoryStream();
pdfDoc.Save(stream);
stream.Position = 0;
//Close the document
pdfDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "DocumentInformation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create a Pdf document
PdfDocument pdfDoc = new PdfDocument();
//Create a Page
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP Page text Schema
PagedTextSchema pagedText = metaData.PagedTextSchema;
//Sets the Page text Schema details of the document
pagedText.MaxPageSize.Width = 500;
pagedText.MaxPageSize.Height = 750;
pagedText.NPages = 1;
pagedText.PlateNames.Add("Sample page");
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
pdfDoc.Save(memoryStream);
//Close the documents
pdfDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DocumentInfor
mation.pdf", "application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("DocumentInformation.pdf",
"application/pdf", memoryStream);
}

```

PDF schema

This schema specifies properties used with Adobe PDF documents.

[PDFSchema](#) class is used to create the PDF Schema. It has the following set of properties.

C#

```
//Create a PDF document
PdfDocument pdfDoc = new PdfDocument();
//Add a page
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP PDF Schema
PDFSchema pdfSchema = metaData.PDFSchema;
//Set the PDF Schema details of the document
pdfSchema.Producer = "Syncfusion";
pdfSchema.PDFVersion = "1.5";
pdfSchema.Keywords = "Essential PDF";
//Save and Close the document
pdfDoc.Save("DocumentInformation.pdf");
pdfDoc.Close(true);
```

VB.NET

```
'Create a PDF document
Dim pdfDoc As New PdfDocument()
'Add a page
Dim page As PdfPage = pdfDoc.Pages.Add()
'Gets XMP object
Dim metaData As XmpMetadata = pdfDoc.DocumentInformation.XmpMetadata
'XMP PDF Schema
Dim pdfSchema As PDFSchema = metaData.PDFSchema
'Set the PDF Schema details of the document
pdfSchema.Producer = "Syncfusion"
pdfSchema.PDFVersion = "1.5"
pdfSchema.Keywords = "Essential PDF"
'Save and Close the document
pdfDoc.Save("DocumentInformation.pdf")
pdfDoc.Close(True)
```

UWP

```
//Essential PDF supports Metadata (XMP) only in Windows Forms, WPF, ASP.NET,
ASP.NET MVC, ASP.NET Core and Xamarin platforms
```

ASP.NET CORE

```
//Create a PDF document
PdfDocument pdfDoc = new PdfDocument();
//Add a page
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP PDF Schema
PDFSchema pdfSchema = metaData.PDFSchema;
//Set the PDF Schema details of the document
pdfSchema.Producer = "Syncfusion";
pdfSchema.PDFVersion = "1.5";
pdfSchema.Keywords = "Essential PDF";
```

```
//Save and close the document
MemoryStream stream = new MemoryStream();
pdfDoc.Save(stream);
stream.Position = 0;
//Close the document
pdfDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "DocumentInformation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a PDF document
PdfDocument pdfDoc = new PdfDocument();
//Add a page
PdfPage page = pdfDoc.Pages.Add();
//Gets XMP object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//XMP PDF Schema
PDFSchema pdfSchema = metaData.PDFSchema;
//Set the PDF Schema details of the document
pdfSchema.Producer = "Syncfusion";
pdfSchema.PDFVersion = "1.5";
pdfSchema.Keywords = "Essential PDF";
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
pdfDoc.Save(memoryStream);
//Close the documents
pdfDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DocumentInfor
mation.pdf", "application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("DocumentInformation.pdf",
"application/pdf", memoryStream);
}
```

Custom Schema

A custom schema defines the structure of the customized information records. You can use the [CustomSchema](#) class to:

- Define custom metadata files and,

- Add them to the PDF document

Add the following code to define a custom schema.

C#

```
//Create Pdf document
PdfDocument pdfDoc = new PdfDocument();
PdfPage page = pdfDoc.Pages.Add();
//Get metaData object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//Create custom schema field
CustomSchema customSchema = new CustomSchema(metaData, "custom",
"http://www.syncfusion.com");
customSchema["Author"] = "Syncfusion";
customSchema["creationDate"] = DateTime.Now.ToString();
customSchema["DOCID"] = "SYNCSAM001";
customSchema["Encryption"] = "Standard";
customSchema["Project"] = "Data processing";
//Save the document
pdfDoc.Save("DocumentInformation.pdf");
pdfDoc.Close(true);
```

VB.NET

```
'Create Pdf document
Dim pdfDoc As New PdfDocument()
Dim page As PdfPage = pdfDoc.Pages.Add()
'Get metaData object
Dim metaData As XmpMetadata = pdfDoc.DocumentInformation.XmpMetadata
'Create custom schema field
Dim customSchema As New CustomSchema(metaData, "custom",
"http://www.syncfusion.com")
customSchema("Author") = "Syncfusion"
customSchema("creationDate") = DateTime.Now.ToString()
customSchema("DOCID") = "SYNCSAM001"
customSchema("Encryption") = "Standard"
customSchema("Project") = "Data processing"
'Save the document
pdfDoc.Save("DocumentInformation.pdf")
pdfDoc.Close(True)
```

UWP

```
//Essential PDF supports Metadata (XMP) only in Windows Forms, WPF, ASP.NET,
ASP.NET MVC, ASP.NET Core and Xamarin platforms
```

ASP.NET CORE

```
//Create Pdf document
PdfDocument pdfDoc = new PdfDocument();
PdfPage page = pdfDoc.Pages.Add();
//Get metaData object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//Create custom schema field
```

```

CustomSchema customSchema = new CustomSchema(metaData, "custom",
"http://www.syncfusion.com");
customSchema["creationDate"] = DateTime.Now.ToString();
customSchema["DOCID"] = "SYNCSAM001";
customSchema["Encryption"] = "Standard";
customSchema["Project"] = "Data processing";
//Save and close the document
MemoryStream stream = new MemoryStream();
pdfDoc.Save(stream);
stream.Position = 0;
//Close the document
pdfDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "DocumentInformation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create Pdf document
PdfDocument pdfDoc = new PdfDocument();
PdfPage page = pdfDoc.Pages.Add();
//Get metaData object
XmpMetadata metaData = pdfDoc.DocumentInformation.XmpMetadata;
//Create custom schema field
CustomSchema customSchema = new CustomSchema(metaData, "custom",
"http://www.syncfusion.com");
customSchema["creationDate"] = DateTime.Now.ToString();
customSchema["DOCID"] = "SYNCSAM001";
customSchema["Encryption"] = "Standard";
customSchema["Project"] = "Data processing";
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
pdfDoc.Save(memoryStream);
//Close the documents
pdfDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DocumentInfor
mation.pdf", "application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("DocumentInformation.pdf",
"application/pdf", memoryStream);
}

```


Adding Custom Schema to the PDF document

Essential PDF allows you to add required metadata (custom schema) to a PDF document.

You can add custom schema using [XmpMetadata](#) class. The following code illustrates this.

C#

```
//Create PDF document
PdfDocument pdfDoc = new PdfDocument();
PdfPage page = pdfDoc.Pages.Add();
//Create XML Document container
XmpMetadata metaData = new
XmpMetadata(pdfDoc.DocumentInformation.XmpMetadata.XmlData);
//Create custom schema
CustomSchema customSchema = new CustomSchema(metaData, "custom",
"http://www.syncfusion.com");
customSchema["Author"] = "Syncfusion";
customSchema["creationDate"] = DateTime.Now.ToString();
customSchema["DOCID"] = "SYNCSAM001";
//Save and close the document
pdfDoc.Save("CustomMetaField.pdf");
pdfDoc.Close(true);
```

VB.NET

```
'Create PDF document
Dim pdfDoc As New PdfDocument()
Dim page As PdfPage = pdfDoc.Pages.Add()
'Create XML Document container
Dim metaData As New
XmpMetadata(pdfDoc.DocumentInformation.XmpMetadata.XmlData)
'Create custom schema
Dim customSchema As New CustomSchema(metaData, "custom",
"http://www.syncfusion.com")
customSchema("Author") = "Syncfusion"
customSchema("creationDate") = DateTime.Now.ToString()
customSchema("DOCID") = "SYNCSAM001"
'Save and close the document
pdfDoc.Save("CustomMetaField.pdf")
pdfDoc.Close(True)
```

UWP

```
//Essential PDF supports Metadata (XMP) only in Windows Forms, WPF, ASP.NET,
ASP.NET MVC, ASP.NET Core and Xamarin platforms
```

ASP.NET CORE

```
//Create PDF document
PdfDocument pdfDoc = new PdfDocument();
PdfPage page = pdfDoc.Pages.Add();
//Create XML Document container
XmpMetadata metaData = new
XmpMetadata(pdfDoc.DocumentInformation.XmpMetadata.XmlData);
//Create custom schema
```

```

CustomSchema customSchema = new CustomSchema(metaData, "custom",
"http://www.syncfusion.com");
customSchema["Author"] = "Syncfusion";
customSchema["creationDate"] = DateTime.Now.ToString();
customSchema["DOCID"] = "SYNCSAM001";
//Save and close the document
MemoryStream stream = new MemoryStream();
pdfDoc.Save(stream);
stream.Position = 0;
//Close the document
pdfDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "CustomMetaField.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Create PDF document
PdfDocument pdfDoc = new PdfDocument();
PdfPage page = pdfDoc.Pages.Add();
//Create XML Document container
XmpMetadata metaData = new
XmpMetadata(pdfDoc.DocumentInformation.XmpMetadata.XmlData);
//Create custom schema
CustomSchema customSchema = new CustomSchema(metaData, "custom",
"http://www.syncfusion.com");
customSchema["Author"] = "Syncfusion";
customSchema["creationDate"] = DateTime.Now.ToString();
customSchema["DOCID"] = "SYNCSAM001";
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
pdfDoc.Save(memoryStream);
//Close the documents
pdfDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("CustomMetaFie
ld.pdf", "application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("CustomMetaField.pdf",
"application/pdf", memoryStream);
}

```

Adding Custom Metadata to the PDF document

The custom metadata can be added in PDF document by using the [CustomMetadata](#) property. Refer to the following code.

C#

```
//Create PDF document
PdfDocument pdfDoc = new PdfDocument();
//Add new PDF page
PdfPage page = pdfDoc.Pages.Add();
//Add Custom MetaData
pdfDoc.DocumentInformation.CustomMetadata["ID"] = "IO1";
pdfDoc.DocumentInformation.CustomMetadata["CompanyName"] = "Syncfusion";
pdfDoc.DocumentInformation.CustomMetadata["Key"] = "DocumentKey";
//Save and close the document
pdfDoc.Save("AddCustomField.pdf");
pdfDoc.Close(true);
```

VB.NET

```
'Create PDF document
Dim pdfDoc As New PdfDocument()
'Add new PDF page
Dim page As PdfPage = pdfDoc.Pages.Add()
'Add Custom MetaData
pdfDoc.DocumentInformation.CustomMetadata("ID") = "IO1"
pdfDoc.DocumentInformation.CustomMetadata("CompanyName") = "Syncfusion"
pdfDoc.DocumentInformation.CustomMetadata("Key") = "DocumentKey"
'Save and close the document
pdfDoc.Save("AddCustomField.pdf")
pdfDoc.Close(True)
```

UWP

```
//Essential PDF supports Metadata (XMP) only in Windows Forms, WPF, ASP.NET,
ASP.NET MVC, ASP.NET Core and Xamarin platforms
```

ASP.NET CORE

```
//Create PDF document
PdfDocument pdfDoc = new PdfDocument();
//Add new PDF page
PdfPage page = pdfDoc.Pages.Add();
//Add Custom MetaData
pdfDoc.DocumentInformation.CustomMetadata["ID"] = "IO1";
pdfDoc.DocumentInformation.CustomMetadata["CompanyName"] = "Syncfusion";
pdfDoc.DocumentInformation.CustomMetadata["Key"] = "DocumentKey";
//Save and close the document
MemoryStream stream = new MemoryStream();
pdfDoc.Save(stream);
stream.Position = 0;
//Close the document
pdfDoc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
```

```
//Define the file name
string fileName = "AddCustomField.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create PDF document
PdfDocument pdfDoc = new PdfDocument();
//Add new PDF page
PdfPage page = pdfDoc.Pages.Add();
//Add Custom MetaData
pdfDoc.DocumentInformation.CustomMetadata["ID"] = "IO1";
pdfDoc.DocumentInformation.CustomMetadata["CompanyName"] = "Syncfusion";
pdfDoc.DocumentInformation.CustomMetadata["Key"] = "DocumentKey";
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
pdfDoc.Save(memoryStream);
//Close the documents
pdfDoc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("AddCustomFiel
d.pdf", "application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("AddCustomField.pdf",
"application/pdf", memoryStream);
}
```

Removing Custom Metadata from an existing PDF document

You can remove the custom metadata from an existing PDF document as follows.

C#

```
//Load the document
PdfLoadedDocument loadedDocument = new PdfLoadedDocument("Input.pdf");
//Remove custom metadata using key name
loadedDocument.DocumentInformation.CustomMetadata.Remove("Key");
//Save and close the document
loadedDocument.Save("Output.pdf");
loadedDocument.Close(true);
```

VB.NET

```
'Load the document
Dim loadedDocument As PdfLoadedDocument = New PdfLoadedDocument("Input.pdf")
'Remove custom metadata using key name
```

```
loadedDocument.DocumentInformation.CustomMetadata.Remove("Key")
'Save and close the document
loadedDocument.Save("Output.pdf")
loadedDocument.Close(True)
```

UWP

```
//Essential PDF supports Metadata (XMP) only in Windows Forms, WPF, ASP.NET,
ASP.NET MVC, ASP.NET Core and Xamarin platforms
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Remove custom metadata using key name
loadedDocument.DocumentInformation.CustomMetadata.Remove("Key");
//Save and close the document
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Close the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Input.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Remove custom metadata using key name
loadedDocument.DocumentInformation.CustomMetadata.Remove("Key");
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
loadedDocument.Save(memoryStream);
//Close the documents
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
```

```

}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}

```

Working with image metadata

Image metadata is a data that describes the characteristics or properties of an image which is embedded in the image file.

Adding XMP metadata along with an image in a PDF document

You can extract the XMP metadata along with an image and add it with an image to the PDF document as shown in the code snippet below.

C#

```

//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image file from the disk which contains XMP metadata
PdfBitmap image = new PdfBitmap("Autumn Leaves.jpg", true);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the document
doc.Save("Output.pdf");
//Close the document
doc.Close(true);

```

VB.NET

```

'Create a new PDF document
Dim doc As New PdfDocument()
'Add a page to the document
Dim page As PdfPage = doc.Pages.Add()
'Create PDF graphics for the page
Dim graphics As PdfGraphics = page.Graphics
'Load the image file from the disk which contains XMP metadata
Dim image As New PdfBitmap("Autumn Leaves.jpg", True)
'Draw the image
graphics.DrawImage(image, 0, 0)
'Save the document
doc.Save("Output.pdf")
'Close the document
doc.Close(True)

```

UWP

```

//Essential PDF supports adding XMP metadata along with an image only in
Windows Forms, WPF, ASP.NET, ASP.NET MVC, ASP.NET Core and Xamarin platforms

```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image as stream which contains XMP metadata
FileStream imageStream = new FileStream("Autumn Leaves.jpg", FileMode.Open,
FileAccess.Read);
PdfBitmap image = new PdfBitmap(imageStream, true);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Creating the stream object
MemoryStream stream = new MemoryStream();
//Save the document as stream
doc.Save(stream);
stream.Position = 0;
//Close the document
doc.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document
PdfDocument doc = new PdfDocument();
//Add a page to the document
PdfPage page = doc.Pages.Add();
//Create PDF graphics for the page
PdfGraphics graphics = page.Graphics;
//Load the image as stream which contains XMP metadata
Stream imageStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Autumn Leaves.jpg");
PdfBitmap image = new PdfBitmap(imageStream, true);
//Draw the image
graphics.DrawImage(image, 0, 0);
//Save the document as stream
MemoryStream stream = new MemoryStream();
doc.Save(stream);
//Close the document
doc.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

Extracting XMP metadata from PDF image

To extract the [XmpMetadata](#) from an image in an existing PDF document, you can use the [ImagesInfo](#) property in the [PdfPageBase](#) class.

Refer to the following code snippet to extract the image metadata from a PDF image.

C#

```
//Load an existing PDF
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Load the first page
PdfPageBase pageBase = loadedDocument.Pages[0];
//Extracts all the images info from first page
PdfImageInfo[] imagesInfo= pageBase.ImagesInfo;
//Extracts the XMP metadata from PDF image
XmpMetadata metadata = imagesInfo[0].XmpMetadata;
//Close the document
loadedDocument.Close(true);
```

VB.NET

```
'Load an existing PDF
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Load the first page
Dim pageBase As PdfPageBase = loadedDocument.Pages(0)
'Extracts all the images info from first page
Dim imagesInfo As PdfImageInfo[] = pageBase.ImagesInfo
'Extracts the XMP metadata from PDF image
XmpMetadata metadata = imagesInfo[0].XmpMetadata;
'Close the document
loadedDocument.Close(True)
```

UWP

```
//PDF supports extracting the image information from PDF document only in
Windows Forms, WPF, ASP.NET, ASP.NET Core and ASP.NET MVC platforms
```

ASP.NET CORE

```
//Load an existing PDF
FileStream docStream = new FileStream("Input.pdf", FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Load the first page
PdfPageBase pageBase = loadedDocument.Pages[0];
//Extracts all the images info from first page
```



```
PdfImageInfo[] imagesInfo= pageBase.GetImagesInfo();
//Extracts the XMP metadata from PDF image
XmpMetadata metadata = imagesInfo[0].XmpMetadata;
//Close the document
loadedDocument.Close(true);
```

XAMARIN

```
//PDF supports extracting the image information from PDF document only in
Windows Forms, WPF, ASP.NET, ASP.NET Core and ASP.NET MVC platforms
```

Note: To extract the image information from PDF page in .NET Core, you need to include [Syncfusion.Pdf.Imaging.Portable](#) assembly reference in .NET Core project.

Working with Color Spaces

Essential PDF allows you to set the color spaces in the following different ways.

- Document Color Space
- Graphics Color Space

Working with color space in document

You can set the color space by using [ColorSpace](#) property in PDF document.

It supports the following types

- Device color Spaces
- CIE-based Color Spaces
- ICC-based Color Spaces

Device Color Space

Device color space simply describes the range of colors that a camera can see, a printer can print, or a monitor can display. These color spaces depend upon the device where it is displayed. It contains the following types.

- DeviceGray
- DeviceRGB
- DeviceCMYK

CIE-based Color Spaces

CIE-based color space in the PDF document is classified as,

- CalGray
- CalRGB
- Lab

You can draw a rectangle on new PDF document with **CalGray** brush using [PdfCalGrayColorSpace](#) class. The following code snippet explains this.

C#

```

//Creates a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Adds a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
//Acquires graphics of the page.
PdfGraphics graphics = pdfPage.Graphics;
//Creates CalGray color space.
PdfCalGrayColorSpace calGrayColorSpace = new PdfCalGrayColorSpace();
//Updates color values.
calGrayColorSpace.Gamma = 0.7;
calGrayColorSpace.WhitePoint = new double[] { 0.2, 1, 0.8 };
PdfCalGrayColor calGrayColorSpace1 = new PdfCalGrayColor(calGrayColorSpace);
calGrayColorSpace1.Gray = 0.1;
PdfBrush brush = new PdfSolidBrush(calGrayColorSpace1);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush
graphics.DrawRectangle(brush, bounds);
//Saves the document.
pdfDocument.Save("Output.pdf");
//closes the document
pdfDocument.Close(true);

```

VB.NET

```

'Creates a new PDF document.
Dim pdfDocument As New PdfDocument()
'Adds a page to the PDF document
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
'Acquires graphics of the page.
Dim graphics As PdfGraphics = pdfPage.Graphics
'Creates CalGray color space.
Dim calGrayColorSpace As New PdfCalGrayColorSpace()
'Updates color values.
calGrayColorSpace.Gamma = 0.7
calGrayColorSpace.WhitePoint = New Double() {0.2, 1, 0.8}
Dim calGrayColorSpace1 As New PdfCalGrayColor(calGrayColorSpace)
calGrayColorSpace1.Gray = 0.1
Dim brush As PdfBrush = New PdfSolidBrush(calGrayColorSpace1)
Dim bounds As New RectangleF(0, 0, 300, 300)
'Draws rectangle using the PdfBrush
graphics.DrawRectangle(brush, bounds)
'Saves the document.
pdfDocument.Save("Output.pdf")
'closes the document
pdfDocument.Close(True)

```

UWP

```

//Creates a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Adds a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
//Acquires graphics of the page.
PdfGraphics graphics = pdfPage.Graphics;
//Creates CalGray color space.
PdfCalGrayColorSpace calGrayColorSpace = new PdfCalGrayColorSpace();

```

```
//Updates color values.
calGrayColorSpace.Gamma = 0.7;
calGrayColorSpace.WhitePoint = new double[] { 0.2, 1, 0.8 };
PdfCalGrayColor calGrayColorSpace1 = new PdfCalGrayColor(calGrayColorSpace);
calGrayColorSpace1.Gray = 0.1;
PdfBrush brush = new PdfSolidBrush(calGrayColorSpace1);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush
graphics.DrawRectangle(brush, bounds);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
//Close the document
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Creates a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Adds a page to the PDF document
PdfPage pdfPage = pdfDocument.Pages.Add();
//Acquires graphics of the page.
PdfGraphics graphics = pdfPage.Graphics;
//Creates CalGray color space.
PdfCalGrayColorSpace calGrayColorSpace = new PdfCalGrayColorSpace();
//Updates color values.
calGrayColorSpace.Gamma = 0.7;
calGrayColorSpace.WhitePoint = new double[] { 0.2, 1, 0.8 };
PdfCalGrayColor calGrayColorSpace1 = new PdfCalGrayColor(calGrayColorSpace);
calGrayColorSpace1.Gray = 0.1;
PdfBrush brush = new PdfSolidBrush(calGrayColorSpace1);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush
graphics.DrawRectangle(brush, bounds);
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Closes the document
pdfDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Adds a page to the PDF document
```

```

PdfPage pdfPage = pdfDocument.Pages.Add();
//Acquires graphics of the page.
PdfGraphics graphics = pdfPage.Graphics;
//Creates CalGray color space.
PdfCalGrayColorSpace calGrayColorSpace = new PdfCalGrayColorSpace();
//Updates color values.
calGrayColorSpace.Gamma = 0.7;
calGrayColorSpace.WhitePoint = new double[] { 0.2, 1, 0.8 };
PdfCalGrayColor calGrayColorSpace1 = new PdfCalGrayColor(calGrayColorSpace);
calGrayColorSpace1.Gray = 0.1;
PdfBrush brush = new PdfSolidBrush(calGrayColorSpace1);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush
graphics.DrawRectangle(brush, bounds);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Close the document.
pdfDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following code example illustrates how to draw a rectangle with **CalGray** brush in existing PDF document.

C#

```

//Loads the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Loads the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Acquires graphics of the page.
PdfGraphics graphics = loadedPage.Graphics;
//Creates CalGray color space.
PdfCalGrayColorSpace calGrayColorSpace = new PdfCalGrayColorSpace();
//Updates color values.
calGrayColorSpace.Gamma = 0.7;
calGrayColorSpace.WhitePoint = new double[] { 0.2, 1, 0.8 };
PdfCalGrayColor calGrayColorSpace1 = new PdfCalGrayColor(calGrayColorSpace);
calGrayColorSpace1.Gray = 0.1;
PdfBrush brush = new PdfSolidBrush(calGrayColorSpace1);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush

```

```
graphics.DrawRectangle(brush, bounds);
//Saves the modified document.
loadedDocument.Save("Output.pdf");
//Closes the document
loadedDocument.Close(true);
```

VB.NET

```
'Loads the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Loads the page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Acquires graphics of the page.
Dim graphics As PdfGraphics = loadedPage.Graphics
'Creates CalGray color space.
Dim calGrayColorSpace As New PdfCalGrayColorSpace()
'Updates color values.
calGrayColorSpace.Gamma = 0.7
calGrayColorSpace.WhitePoint = New Double() {0.2, 1, 0.8}
Dim calGrayColorSpace1 As New PdfCalGrayColor(calGrayColorSpace)
calGrayColorSpace1.Gray = 0.1
Dim brush As PdfBrush = New PdfSolidBrush(calGrayColorSpace1)
Dim bounds As New RectangleF(0, 0, 300, 300)
'Draws rectangle by using the PdfBrush
graphics.DrawRectangle(brush, bounds)
'Saves the modified document.
loadedDocument.Save("Output.pdf")
'closes the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Loads the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Acquires graphics of the page.
PdfGraphics graphics = loadedPage.Graphics;
//Creates CalGray color space.
PdfCalGrayColorSpace calGrayColorSpace = new PdfCalGrayColorSpace();
//Updates color values.
calGrayColorSpace.Gamma = 0.7;
calGrayColorSpace.WhitePoint = new double[] { 0.2, 1, 0.8 };
PdfCalGrayColor calGrayColorSpace1 = new PdfCalGrayColor(calGrayColorSpace);
calGrayColorSpace1.Gray = 0.1;
PdfBrush brush = new PdfSolidBrush(calGrayColorSpace1);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
```

```
//Draws rectangle by using the PdfBrush
graphics.DrawRectangle(brush, bounds);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Loads the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Acquires graphics of the page.
PdfGraphics graphics = loadedPage.Graphics;
//Creates CalGray color space.
PdfCalGrayColorSpace calGrayColorSpace = new PdfCalGrayColorSpace();
//Updates color values.
calGrayColorSpace.Gamma = 0.7;
calGrayColorSpace.WhitePoint = new double[] { 0.2, 1, 0.8 };
PdfCalGrayColor calGrayColorSpace1 = new PdfCalGrayColor(calGrayColorSpace);
calGrayColorSpace1.Gray = 0.1;
PdfBrush brush = new PdfSolidBrush(calGrayColorSpace1);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush
graphics.DrawRectangle(brush, bounds);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Closes the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Loads the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Acquires graphics of the page.
```

```

PdfGraphics graphics = loadedPage.Graphics;
//Creates CalGray color space.
PdfCalGrayColorSpace calGrayColorSpace = new PdfCalGrayColorSpace();
//Updates color values.
calGrayColorSpace.Gamma = 0.7;
calGrayColorSpace.WhitePoint = new double[] { 0.2, 1, 0.8 };
PdfCalGrayColor calGrayColorSpace1 = new PdfCalGrayColor(calGrayColorSpace);
calGrayColorSpace1.Gray = 0.1;
PdfBrush brush = new PdfSolidBrush(calGrayColorSpace1);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush
graphics.DrawRectangle(brush, bounds);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

ICC-based Color Spaces

To create color for brush/pen to render shape or text in the PDF document, you can use ICC based color spaces.

It contains the following types:

- Special color spaces –Pantone colors
- Indexed
- Separation

The following code example illustrates how to set the indexed ICC color space using [PdfCalRGBColorSpace](#) class in new PDF document.

C#

```

//Creates a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Adds a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
//Acquires graphics of the page.
PdfGraphics graphics = pdfPage.Graphics;
//Creates ICCBased color space.

```

```

PdfCalRGBColorSpace calRgbCS = new PdfCalRGBColorSpace();
calRgbCS.Gamma = new double[] { 7.6, 5.1, 8.5 };
calRgbCS.Matrix = new double[] { 1, 0, 0, 0, 1, 0, 0, 0, 1 };
calRgbCS.WhitePoint = new double[] { 0.7, 1, 0.8 };
//Reads the ICC profile.
FileStream fileStream = new FileStream(@"input.icc", FileMode.Open,
FileAccess.Read);
byte[] profileData = new byte[fileStream.Length];
fileStream.Read(profileData, 0, profileData.Length);
fileStream.Close();
//Instantiates ICC color space.
PdfICCColorSpace iccBasedCS = new PdfICCColorSpace();
iccBasedCS.ProfileData = profileData;
iccBasedCS.AlternateColorSpace = calRgbCS;
iccBasedCS.ColorComponents = 3;
iccBasedCS.Range = new double[] { 0.0, 1.0, 0.0, 1.0, 0.0, 1.0 };
PdfICCColor red = new PdfICCColor(iccBasedCS);
red.ColorComponents = new double[] { 1, 0, 1 };
PdfBrush brush = new PdfSolidBrush(red);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush.
graphics.DrawRectangle(brush, bounds);
//Saves the document.
pdfDocument.Save("Output.pdf");
//Closes the document
pdfDocument.Close(true);

```

VB.NET

```

'Creates a new PDF document.
Dim pdfDocument As New PdfDocument()
'Adds a page to the PDF document.
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
'Acquires graphics of the page.
Dim graphics As PdfGraphics = pdfPage.Graphics
'Creates ICCBased color space.
Dim calRgbCS As New PdfCalRGBColorSpace()
calRgbCS.Gamma = New Double() {7.6, 5.1, 8.5}
calRgbCS.Matrix = New Double() {1, 0, 0, 0, 1, 0, 0, 0, 1}
calRgbCS.WhitePoint = New Double() {0.7, 1, 0.8}
'Reads the ICC profile.
Dim fileStream As New FileStream("input.icc", FileMode.Open,
FileAccess.Read)
Dim profileData(fileStream.Length - 1) As Byte
fileStream.Read(profileData, 0, profileData.Length)
fileStream.Close()
'Instantiates ICC color space.
Dim iccBasedCS As New PdfICCColorSpace()
iccBasedCS.ProfileData = profileData
iccBasedCS.AlternateColorSpace = calRgbCS
iccBasedCS.ColorComponents = 3
iccBasedCS.Range = New Double() {0.0, 1.0, 0.0, 1.0, 0.0, 1.0}
Dim red As New PdfICCColor(iccBasedCS)
red.ColorComponents = New Double() {1, 0, 1}
Dim brush As PdfBrush = New PdfSolidBrush(red)
Dim bounds As New RectangleF(0, 0, 300, 300)

```



```
'Draws rectangle by using the PdfBrush.
graphics.DrawRectangle(brush, bounds)
'Saves the document.
pdfDocument.Save("Output.pdf")
'Closes the document
pdfDocument.Close(True)
```

UWP

```
//Creates a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Adds a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
//Acquires graphics of the page.
PdfGraphics graphics = pdfPage.Graphics;
//Creates ICCBased color space.
PdfCalRGBColorSpace calRgbCS = new PdfCalRGBColorSpace();
calRgbCS.Gamma = new double[] { 7.6, 5.1, 8.5 };
calRgbCS.Matrix = new double[] { 1, 0, 0, 0, 1, 0, 0, 0, 1 };
calRgbCS.WhitePoint = new double[] { 0.7, 1, 0.8 };
//Reads the ICC profile.
Stream fileStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.icc");
byte[] profileData = new byte[fileStream.Length];
fileStream.Read(profileData, 0, profileData.Length);
fileStream.Dispose();
//Instantiates ICC color space.
PdfICCColorSpace iccBasedCS = new PdfICCColorSpace();
iccBasedCS.ProfileData = profileData;
iccBasedCS.AlternateColorSpace = calRgbCS;
iccBasedCS.ColorComponents = 3;
iccBasedCS.Range = new double[] { 0.0, 1.0, 0.0, 1.0, 0.0, 1.0 };
PdfICCColor red = new PdfICCColor(iccBasedCS);
red.ColorComponents = new double[] { 1, 0, 1 };
PdfBrush brush = new PdfSolidBrush(red);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush.
graphics.DrawRectangle(brush, bounds);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
//Close the document
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");
```

ASP.NET CORE

```
//Creates a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Adds a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
//Acquires graphics of the page.
PdfGraphics graphics = pdfPage.Graphics;
```

```

//Creates ICCBased color space.
PdfCalRGBColorSpace calRgbCS = new PdfCalRGBColorSpace();
calRgbCS.Gamma = new double[] { 7.6, 5.1, 8.5 };
calRgbCS.Matrix = new double[] { 1, 0, 0, 0, 1, 0, 0, 0, 1 };
calRgbCS.WhitePoint = new double[] { 0.7, 1, 0.8 };
//Reads the ICC profile.
FileStream fileStream = new FileStream(@"input.icc", FileMode.Open,
FileAccess.Read);
byte[] profileData = new byte[fileStream.Length];
fileStream.Read(profileData, 0, profileData.Length);
fileStream.Close();
//Instantiates ICC color space.
PdfICCColorSpace iccBasedCS = new PdfICCColorSpace();
iccBasedCS.ProfileData = profileData;
iccBasedCS.AlternateColorSpace = calRgbCS;
iccBasedCS.ColorComponents = 3;
iccBasedCS.Range = new double[] { 0.0, 1.0, 0.0, 1.0, 0.0, 1.0 };
PdfICCColor red = new PdfICCColor(iccBasedCS);
red.ColorComponents = new double[] { 1, 0, 1 };
PdfBrush brush = new PdfSolidBrush(red);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush.
graphics.DrawRectangle(brush, bounds);
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Closes the document
pdfDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Adds a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
//Acquires graphics of the page.
PdfGraphics graphics = pdfPage.Graphics;
//Creates ICCBased color space.
PdfCalRGBColorSpace calRgbCS = new PdfCalRGBColorSpace();
calRgbCS.Gamma = new double[] { 7.6, 5.1, 8.5 };
calRgbCS.Matrix = new double[] { 1, 0, 0, 0, 1, 0, 0, 0, 1 };
calRgbCS.WhitePoint = new double[] { 0.7, 1, 0.8 };
//Reads the ICC profile.
Stream fileStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
input.icc");
byte[] profileData = new byte[fileStream.Length];
fileStream.Read(profileData, 0, profileData.Length);

```

```

fileStream.Close();
//Instantiates ICC color space.
PdfICCColorSpace iccBasedCS = new PdfICCColorSpace();
iccBasedCS.ProfileData = profileData;
iccBasedCS.AlternateColorSpace = calRgbCS;
iccBasedCS.ColorComponents = 3;
iccBasedCS.Range = new double[] { 0.0, 1.0, 0.0, 1.0, 0.0, 1.0 };
PdfICCColor red = new PdfICCColor(iccBasedCS);
red.ColorComponents = new double[] { 1, 0, 1 };
PdfBrush brush = new PdfSolidBrush(red);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush.
graphics.DrawRectangle(brush, bounds);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Close the document.
pdfDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

The following code example illustrates how to set the indexed ICC color space in existing PDF document.

C#

```

//Loads the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Loads the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Acquires graphics of the page.
PdfGraphics graphics = loadedPage.Graphics;
//Creates ICCBased color space.
PdfCalRGBColorSpace calRgbCS = new PdfCalRGBColorSpace();
calRgbCS.Gamma = new double[] { 7.6, 5.1, 8.5 };
calRgbCS.Matrix = new double[] { 1, 0, 0, 0, 1, 0, 0, 0, 1 };
calRgbCS.WhitePoint = new double[] { 0.7, 1, 0.8 };
//Reads the ICC profile.
FileStream fileStream = new FileStream(@"input.icc", FileMode.Open,
FileAccess.Read);
byte[] profileData = new byte[fileStream.Length];
fileStream.Read(profileData, 0, profileData.Length);
fileStream.Close();
//Instantiates ICC color space.

```

```

PdfICCColorSpace iccBasedCS = new PdfICCColorSpace();
iccBasedCS.ProfileData = profileData;
iccBasedCS.AlternateColorSpace = calRgbCS;
iccBasedCS.ColorComponents = 3;
iccBasedCS.Range = new double[] { 0.0, 1.0, 0.0, 1.0, 0.0, 1.0 };
PdfICCColor red = new PdfICCColor(iccBasedCS);
red.ColorComponents = new double[] { 1, 0, 1 };
PdfBrush brush = new PdfSolidBrush(red);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush.
graphics.DrawRectangle(brush, bounds);
//Saves the document.
loadedDocument.Save("Output.pdf");
//Closes the document
loadedDocument.Close(true);

```

VB.NET

```

'Loads the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Loads the page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0), PdfLoadedPage)
'Acquires graphics of the page.
Dim graphics As PdfGraphics = loadedPage.Graphics
'Creates CalGray color space.
Dim calGrayColorSpace As New PdfCalGrayColorSpace()
'Updates color values.
calGrayColorSpace.Gamma = 0.7
calGrayColorSpace.WhitePoint = New Double() {0.2, 1, 0.8}
Dim calGrayColorSpace1 As New PdfCalGrayColor(calGrayColorSpace)
calGrayColorSpace1.Gray = 0.1
Dim brush As PdfBrush = New PdfSolidBrush(calGrayColorSpace1)
Dim bounds As New RectangleF(0, 0, 300, 300)
'Draws rectangle by using the PdfBrush
graphics.DrawRectangle(brush, bounds)
'Saves the document.
loadedDocument.Save("Output.pdf")
'Closes the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Loads the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Acquires graphics of the page.

```

```

PdfGraphics graphics = loadedPage.Graphics;
//Creates ICCBased color space.
PdfCalRGBColorSpace calRgbCS = new PdfCalRGBColorSpace();
calRgbCS.Gamma = new double[] { 7.6, 5.1, 8.5 };
calRgbCS.Matrix = new double[] { 1, 0, 0, 0, 1, 0, 0, 0, 1 };
calRgbCS.WhitePoint = new double[] { 0.7, 1, 0.8 };
//Reads the ICC profile.
Stream fileStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.Data.input.icc");
byte[] profileData = new byte[fileStream.Length];
fileStream.Read(profileData, 0, profileData.Length);
fileStream.Dispose();
//Instantiates ICC color space.
PdfICColorSpace iccBasedCS = new PdfICColorSpace();
iccBasedCS.ProfileData = profileData;
iccBasedCS.AlternateColorSpace = calRgbCS;
iccBasedCS.ColorComponents = 3;
iccBasedCS.Range = new double[] { 0.0, 1.0, 0.0, 1.0, 0.0, 1.0 };
PdfICColor red = new PdfICColor(iccBasedCS);
red.ColorComponents = new double[] { 1, 0, 1 };
PdfBrush brush = new PdfSolidBrush(red);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush.
graphics.DrawRectangle(brush, bounds);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Loads the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Acquires graphics of the page.
PdfGraphics graphics = loadedPage.Graphics;
//Creates ICCBased color space.
PdfCalRGBColorSpace calRgbCS = new PdfCalRGBColorSpace();
calRgbCS.Gamma = new double[] { 7.6, 5.1, 8.5 };
calRgbCS.Matrix = new double[] { 1, 0, 0, 0, 1, 0, 0, 0, 1 };
calRgbCS.WhitePoint = new double[] { 0.7, 1, 0.8 };
//Reads the ICC profile.
FileStream fileStream = new FileStream(@"input.icc", FileMode.Open,
FileAccess.Read);
byte[] profileData = new byte[fileStream.Length];
fileStream.Read(profileData, 0, profileData.Length);
fileStream.Close();
//Instantiates ICC color space.

```

```

PdfICCColorSpace iccBasedCS = new PdfICCColorSpace();
iccBasedCS.ProfileData = profileData;
iccBasedCS.AlternateColorSpace = calRgbCS;
iccBasedCS.ColorComponents = 3;
iccBasedCS.Range = new double[] { 0.0, 1.0, 0.0, 1.0, 0.0, 1.0 };
PdfICCColor red = new PdfICCColor(iccBasedCS);
red.ColorComponents = new double[] { 1, 0, 1 };
PdfBrush brush = new PdfSolidBrush(red);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush.
graphics.DrawRectangle(brush, bounds);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Closes the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Signature.Assets.Barcode.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Loads the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Acquires graphics of the page.
PdfGraphics graphics = loadedPage.Graphics;
//Creates ICCBased color space.
PdfCalRGBColorSpace calRgbCS = new PdfCalRGBColorSpace();
calRgbCS.Gamma = new double[] { 7.6, 5.1, 8.5 };
calRgbCS.Matrix = new double[] { 1, 0, 0, 0, 1, 0, 0, 0, 1 };
calRgbCS.WhitePoint = new double[] { 0.7, 1, 0.8 };
//Reads the ICC profile.
Stream fileStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Signature.Assets.rgb.icc");
byte[] profileData = new byte[fileStream.Length];
fileStream.Read(profileData, 0, profileData.Length);
fileStream.Close();
//Instantiates ICC color space.
PdfICCColorSpace iccBasedCS = new PdfICCColorSpace();
iccBasedCS.ProfileData = profileData;
iccBasedCS.AlternateColorSpace = calRgbCS;
iccBasedCS.ColorComponents = 3;
iccBasedCS.Range = new double[] { 0.0, 1.0, 0.0, 1.0, 0.0, 1.0 };
PdfICCColor red = new PdfICCColor(iccBasedCS);
red.ColorComponents = new double[] { 1, 0, 1 };

```

```

PdfBrush brush = new PdfSolidBrush(red);
RectangleF bounds = new RectangleF(0, 0, 300, 300);
//Draws rectangle by using the PdfBrush.
graphics.DrawRectangle(brush, bounds);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Pantone colors

The following code example illustrates how to draw the graphics elements by using Pantone colors through [PdfSeparationColorSpace](#) class in new PDF document.

C#

```

// Creates a new document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
// Creates exponential interpolation function
PdfExponentialInterpolationFunction function = new
PdfExponentialInterpolationFunction(true);
float[] numberArray = new float[4];
numberArray[0] = 0.38f;
numberArray[1] = 0.88f;
function.C1 = numberArray;
// Creates SeparationColorSpace
PdfSeparationColorSpace colorSpace = new PdfSeparationColorSpace();
colorSpace.TintTransform = function;
colorSpace.Colorant = "PANTONE Orange 021 C";
PdfSeparationColor color = new PdfSeparationColor(colorSpace);
color.Tint = 0.7;
RectangleF bounds = new RectangleF(20, 70, 200, 100);
PdfPen pen = new PdfPen(color);
//Draws the rectangle
page.Graphics.DrawRectangle(pen, bounds);
//Saves the document
document.Save("SeparationColor.pdf");
//Closes the document
document.Close(true);

```

VB.NET

```

' Creates a new document
Dim document As PdfDocument = New PdfDocument()
' Creates a page
Dim page As PdfPage = document.Pages.Add()
' Creates exponential interpolation function
Dim [function] As PdfExponentialInterpolationFunction = New
PdfExponentialInterpolationFunction(True)
Dim numberArray() As Single = New Single(4) {}
numberArray(0) = 0.38F
numberArray(1) = 0.88F
[function].Cl = numberArray
' Creates SeparationColorSpace
Dim colorSpace As PdfSeparationColorSpace = New PdfSeparationColorSpace()
colorSpace.TintTransform = [function]
colorSpace.Colorant = "PANTONE Orange 021 C"
Dim color As PdfSeparationColor = New PdfSeparationColor(colorSpace)
color.Tint = 0.7
Dim bounds As RectangleF = New RectangleF(20, 70, 200, 100)
Dim pen As PdfPen = New PdfPen(color)
'Draws the rectangle
page.Graphics.DrawRectangle(pen, bounds)
'Saves the document
document.Save("SeparationColor.pdf")
'Closes the document
document.Close(True)

```

UWP

```

// Creates a new document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
// Creates exponential interpolation function
PdfExponentialInterpolationFunction function = new
PdfExponentialInterpolationFunction(true);
float[] numberArray = new float[4];
numberArray[0] = 0.38f;
numberArray[1] = 0.88f;
function.Cl = numberArray;
// Creates SeparationColorSpace
PdfSeparationColorSpace colorSpace = new PdfSeparationColorSpace();
colorSpace.TintTransform = function;
colorSpace.Colorant = "PANTONE Orange 021 C";
PdfSeparationColor color = new PdfSeparationColor(colorSpace);
color.Tint = 0.7;
RectangleF bounds = new RectangleF(20, 70, 200, 100);
PdfPen pen = new PdfPen(color);
//Draws the rectangle
page.Graphics.DrawRectangle(pen, bounds);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await document.SaveAsync(stream);
//Close the document

```



```
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "SeparationColor.pdf");
```

ASP.NET CORE

```
// Creates a new document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
// Creates exponential interpolation function
PdfExponentialInterpolationFunction function = new
PdfExponentialInterpolationFunction(true);
float[] numberArray = new float[4];
numberArray[0] = 0.38f;
numberArray[1] = 0.88f;
function.C1 = numberArray;
// Creates SeparationColorSpace
PdfSeparationColorSpace colorSpace = new PdfSeparationColorSpace();
colorSpace.TintTransform = function;
colorSpace.Colorant = "PANTONE Orange 021 C";
PdfSeparationColor color = new PdfSeparationColor(colorSpace);
color.Tint = 0.7;
RectangleF bounds = new RectangleF(20, 70, 200, 100);
PdfPen pen = new PdfPen(color);
//Draws the rectangle
page.Graphics.DrawRectangle(pen, bounds);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "SeparationColor.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
// Creates a new document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
// Creates exponential interpolation function
PdfExponentialInterpolationFunction function = new
PdfExponentialInterpolationFunction(true);
float[] numberArray = new float[4];
numberArray[0] = 0.38f;
numberArray[1] = 0.88f;
function.C1 = numberArray;
// Creates SeparationColorSpace
```

```

PdfSeparationColorSpace colorSpace = new PdfSeparationColorSpace();
colorSpace.TintTransform = function;
colorSpace.Colorant = "PANTONE Orange 021 C";
PdfSeparationColor color = new PdfSeparationColor(colorSpace);
color.Tint = 0.7;
RectangleF bounds = new RectangleF(20, 70, 200, 100);
PdfPen pen = new PdfPen(color);
//Draws the rectangle
page.Graphics.DrawRectangle(pen, bounds);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document.
document.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("SeparationCol
or.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("SeparationColor.pdf",
"application/pdf", stream);
}

```

The following code example illustrates how to draw the graphics elements by using Pantone colors in existing PDF document.

C#

```

//Loads the document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
// Creates exponential interpolation function
PdfExponentialInterpolationFunction function = new
PdfExponentialInterpolationFunction(true);
float[] numberArray = new float[4];
numberArray[0] = 0.38f;
numberArray[1] = 0.88f;
function.C1 = numberArray;
// Creates SeparationColorSpace
PdfSeparationColorSpace colorSpace = new PdfSeparationColorSpace();
colorSpace.TintTransform = function;
colorSpace.Colorant = "PANTONE Orange 021 C";
PdfSeparationColor color = new PdfSeparationColor(colorSpace);
color.Tint = 0.7;
RectangleF bounds = new RectangleF(20, 70, 200, 100);
PdfPen pen = new PdfPen(color);
//Draws the rectangle
loadedPage.Graphics.DrawRectangle(pen, bounds);
//Saves the document

```

```
loadedDocument.Save("SeparationColor.pdf");
//Closes the document
loadedDocument.Close(true);
```

VB.NET

```
'Loads the document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
' Creates exponential interpolation function
Dim [function] As PdfExponentialInterpolationFunction = New
PdfExponentialInterpolationFunction(True)
Dim numberArray() As Single = New Single(3) {}
numberArray(0) = 0.38F
numberArray(1) = 0.88F
[function].C1 = numberArray
' Creates SeparationColorSpace
Dim colorSpace As PdfSeparationColorSpace = New PdfSeparationColorSpace()
colorSpace.TintTransform = [function]
colorSpace.Colorant = "PANTONE Orange 021 C"
Dim color As PdfSeparationColor = New PdfSeparationColor(colorSpace)
color.Tint = 0.7
Dim bounds As RectangleF = New RectangleF(20, 70, 200, 100)
Dim pen As PdfPen = New PdfPen(color)
'Draws the rectangle
loadedPage.Graphics.DrawRectangle(pen, bounds)
'Saves the document
loadedDocument.Save("SeparationColor.pdf")
'Closes the document
loadedDocument.Close(True)
```

UWP

```
//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
// Creates exponential interpolation function
PdfExponentialInterpolationFunction function = new PdfExponentialInterpolationFunction(true);
float[] numberArray = new float[4];
numberArray[0] = 0.38f;
numberArray[1] = 0.88f;
function.C1 = numberArray;
// Creates SeparationColorSpace
PdfSeparationColorSpace colorSpace = new PdfSeparationColorSpace();
colorSpace.TintTransform = function;
colorSpace.Colorant = "PANTONE Orange 021 C";
```

```

PdfSeparationColor color = new PdfSeparationColor(colorSpace);
color.Tint = 0.7;
RectangleF bounds = new RectangleF(20, 70, 200, 100);
PdfPen pen = new PdfPen(color);
//Draws the rectangle
loadedPage.Graphics.DrawRectangle(pen, bounds);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "SeparationColor.pdf");

```

ASP.NET CORE

```

//Load the PDF document
FileStream docStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
// Creates exponential interpolation function
PdfExponentialInterpolationFunction function = new
PdfExponentialInterpolationFunction(true);
float[] numberArray = new float[4];
numberArray[0] = 0.38f;
numberArray[1] = 0.88f;
function.C1 = numberArray;
// Creates SeparationColorSpace
PdfSeparationColorSpace colorSpace = new PdfSeparationColorSpace();
colorSpace.TintTransform = function;
colorSpace.Colorant = "PANTONE Orange 021 C";
PdfSeparationColor color = new PdfSeparationColor(colorSpace);
color.Tint = 0.7;
RectangleF bounds = new RectangleF(20, 70, 200, 100);
PdfPen pen = new PdfPen(color);
//Draws the rectangle
loadedPage.Graphics.DrawRectangle(pen, bounds);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Closes the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "SeparationColor.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream

```

```

Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
// Creates exponential interpolation function
PdfExponentialInterpolationFunction function = new
PdfExponentialInterpolationFunction(true);
float[] numberArray = new float[4];
numberArray[0] = 0.38f;
numberArray[1] = 0.88f;
function.C1 = numberArray;
// Creates SeparationColorSpace
PdfSeparationColorSpace colorSpace = new PdfSeparationColorSpace();
colorSpace.TintTransform = function;
colorSpace.Colorant = "PANTONE Orange 021 C";
PdfSeparationColor color = new PdfSeparationColor(colorSpace);
color.Tint = 0.7;
RectangleF bounds = new RectangleF(20, 70, 200, 100);
PdfPen pen = new PdfPen(color);
//Draws the rectangle
loadedPage.Graphics.DrawRectangle(pen, bounds);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("SeparationCol
or.pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("SeparationColor.pdf",
"application/pdf", stream);
}

```

Working with color space in graphics

You can set the color spaces to the particular object in the PDF document by using the [ColorSpace](#) property in [PdfGraphics](#) class.

The following code illustrates how to use the color spaces in particular objects in new PDF document.

C#

```

//Creates a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Adds a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
//Acquires graphics of the page.

```

```

PdfGraphics graphics = pdfPage.Graphics;
PdfPen pen = new PdfPen(Color.Red);
PdfBrush brush = new PdfSolidBrush(Color.Blue);
RectangleF rectangle = new RectangleF(0, 0, 100, 100);
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Save();
//GrayScale color space.
graphics.ColorSpace = PdfColorSpace.GrayScale;
graphics.DrawRectangle(pen, brush, rectangle);
//CMYK color space.
graphics.ColorSpace = PdfColorSpace.CMYK;
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Restore();
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
//Draws by using the PdfBrush.
graphics.DrawRectangle(brush, rectangle);
//Saves the document.
pdfDocument.Save("Output.pdf");
//Closes the document
pdfDocument.Close(true);

```

VB.NET

```

'Creates a new PDF document.
Dim pdfDocument As New PdfDocument()
'Adds a page to the PDF document.
Dim pdfPage As PdfPage = pdfDocument.Pages.Add()
'Acquires graphics of the page.
Dim graphics As PdfGraphics = pdfPage.Graphics
Dim pen As New PdfPen(Color.Red)
Dim brush As PdfBrush = New PdfSolidBrush(Color.Blue)
Dim rectangle As New RectangleF(0, 0, 100, 100)
'Default color space.
graphics.DrawRectangle(pen, brush, rectangle)
graphics.Save()
'GrayScale color space.
graphics.ColorSpace = PdfColorSpace.GrayScale
graphics.DrawRectangle(pen, brush, rectangle)
'CMYK color space.
graphics.ColorSpace = PdfColorSpace.CMYK
graphics.DrawRectangle(pen, brush, rectangle)
graphics.Restore()
'Default color space.
graphics.DrawRectangle(pen, brush, rectangle)
'Draws by using the PdfBrush.
graphics.DrawRectangle(brush, rectangle)
'Saves the document.
pdfDocument.Save("Output.pdf")
'Closes the document
pdfDocument.Close(True)

```

UWP

```

//Creates a new PDF document.

```

```

PdfDocument pdfDocument = new PdfDocument();
//Adds a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
//Acquires graphics of the page.
PdfGraphics graphics = pdfPage.Graphics;
PdfPen pen = new PdfPen(new PdfColor(255, 0, 0));
PdfBrush brush = new PdfSolidBrush(new PdfColor(0, 0, 255));
RectangleF rectangle = new RectangleF(0, 0, 100, 100);
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Save();
//GrayScale color space.
graphics.ColorSpace = PdfColorSpace.GrayScale;
graphics.DrawRectangle(pen, brush, rectangle);
//CMYK color space.
graphics.ColorSpace = PdfColorSpace.CMYK;
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Restore();
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
//Draws by using the PdfBrush.
graphics.DrawRectangle(brush, rectangle);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
//Close the document
pdfDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP
section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Creates a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Adds a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
//Acquires graphics of the page.
PdfGraphics graphics = pdfPage.Graphics;
PdfPen pen = new PdfPen(Color.Red);
PdfBrush brush = new PdfSolidBrush(Color.Blue);
RectangleF rectangle = new RectangleF(0, 0, 100, 100);
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Save();
//GrayScale color space.
graphics.ColorSpace = PdfColorSpace.GrayScale;
graphics.DrawRectangle(pen, brush, rectangle);
//CMYK color space.
graphics.ColorSpace = PdfColorSpace.CMYK;
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Restore();
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
//Draws by using the PdfBrush.
graphics.DrawRectangle(brush, rectangle);

```

```
//Save the document into stream
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Closes the document
pdfDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Creates a new PDF document.
PdfDocument pdfDocument = new PdfDocument();
//Adds a page to the PDF document.
PdfPage pdfPage = pdfDocument.Pages.Add();
//Acquires graphics of the page.
PdfGraphics graphics = pdfPage.Graphics;
PdfPen pen = new PdfPen(Syncfusion.Drawing.Color.Red);
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Blue);
RectangleF rectangle = new RectangleF(0, 0, 100, 100);
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Save();
//GrayScale color space.
graphics.ColorSpace = PdfColorSpace.GrayScale;
graphics.DrawRectangle(pen, brush, rectangle);
//CMYK color space.
graphics.ColorSpace = PdfColorSpace.CMYK;
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Restore();
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
//Draws by using the PdfBrush.
graphics.DrawRectangle(brush, rectangle);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
//Close the document.
pdfDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
```



```
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}
```

The following code illustrates how to use the color spaces in particular objects in existing PDF document.

C#

```
//Loads the existing PDF document.
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(fileName);
//Loads the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Acquires graphics of the page.
PdfGraphics graphics = loadedPage.Graphics;
PdfPen pen = new PdfPen(Color.Red);
PdfBrush brush = new PdfSolidBrush(Color.Blue);
RectangleF rectangle = new RectangleF(0, 0, 100, 100);
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Save();
//GrayScale color space.
graphics.ColorSpace = PdfColorSpace.GrayScale;
graphics.DrawRectangle(pen, brush, rectangle);
//CMYK color space.
graphics.ColorSpace = PdfColorSpace.CMYK;
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Restore();
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
//Draws by using the PdfBrush.
graphics.DrawRectangle(brush, rectangle);
//Saves the document.
loadedDocument.Save("Output.pdf");
//Closes the document
loadedDocument.Close(true);
```

VB.NET

```
'Loads the existing PDF document.
Dim loadedDocument As New PdfLoadedDocument(fileName)
'Loads the page
Dim loadedPage As PdfLoadedPage = TryCast(loadedDocument.Pages(0),
PdfLoadedPage)
'Acquires graphics of the page.
Dim graphics As PdfGraphics = loadedPage.Graphics
Dim pen As New PdfPen(Color.Red)
Dim brush As PdfBrush = New PdfSolidBrush(Color.Blue)
Dim rectangle As New RectangleF(0, 0, 100, 100)
'Default color space.
graphics.DrawRectangle(pen, brush, rectangle)
graphics.Save()
'GrayScale color space.
graphics.ColorSpace = PdfColorSpace.GrayScale
graphics.DrawRectangle(pen, brush, rectangle)
'CMYK color space.
graphics.ColorSpace = PdfColorSpace.CMYK
```

```

graphics.DrawRectangle(pen, brush, rectangle)
graphics.Restore()
'Default color space.
graphics.DrawRectangle(pen, brush, rectangle)
'Draws by using the PdfBrush.
graphics.DrawRectangle(brush, rectangle)
'Saves the document.
loadedDocument.Save("Output.pdf")
'Closes the document
loadedDocument.Close(True)

```

UWP

```

//Create the file open picker
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".pdf");
//Browse and chose the file
StorageFile file = await picker.PickSingleFileAsync();
//Creates an empty PDF loaded document instance
PdfLoadedDocument loadedDocument = new PdfLoadedDocument();
//Loads or opens an existing PDF document through Open method of PdfLoadedDocument class
await loadedDocument.OpenAsync(file);
//Loads the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Acquires graphics of the page.
PdfGraphics graphics = loadedPage.Graphics;
PdfPen pen = new PdfPen(new PdfColor(255, 0, 0));
PdfBrush brush = new PdfSolidBrush(new PdfColor(0, 0, 255));
RectangleF rectangle = new RectangleF(0, 0, 100, 100);
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Save();
//GrayScale color space.
graphics.ColorSpace = PdfColorSpace.GrayScale;
graphics.DrawRectangle(pen, brush, rectangle);
//CMYK color space.
graphics.ColorSpace = PdfColorSpace.CMYK;
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Restore();
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
//Draws by using the PdfBrush.
graphics.DrawRectangle(brush, rectangle);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
await loadedDocument.SaveAsync(stream);
//Close the document
loadedDocument.Close(true);
//Save the stream as PDF document file in local machine. Refer to PDF/UWP section for respected code samples
Save(stream, "Output.pdf");

```

ASP.NET CORE

```

//Load the PDF document

```

```

FileStream docStream = new FileStream(fileName, FileMode.Open,
    FileAccess.Read);
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Loads the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Acquires graphics of the page.
PdfGraphics graphics = loadedPage.Graphics;
PdfPen pen = new PdfPen(Color.Red);
PdfBrush brush = new PdfSolidBrush(Color.Blue);
RectangleF rectangle = new RectangleF(0, 0, 100, 100);
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Save();
//GrayScale color space.
graphics.ColorSpace = PdfColorSpace.GrayScale;
graphics.DrawRectangle(pen, brush, rectangle);
//CMYK color space.
graphics.ColorSpace = PdfColorSpace.CMYK;
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Restore();
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
//Draws by using the PdfBrush.
graphics.DrawRectangle(brush, rectangle);
//Save the document into stream
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
stream.Position = 0;
//Closes the document
loadedDocument.Close(true);
//Defining the ContentType for pdf file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Load the file as stream
Stream docStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.Assets.
Sample.pdf");
PdfLoadedDocument loadedDocument = new PdfLoadedDocument(docStream);
//Loads the page
PdfLoadedPage loadedPage = loadedDocument.Pages[0] as PdfLoadedPage;
//Acquires graphics of the page.
PdfGraphics graphics = loadedPage.Graphics;
PdfPen pen = new PdfPen(Syncfusion.Drawing.Color.Red);
PdfBrush brush = new PdfSolidBrush(Syncfusion.Drawing.Color.Blue);
RectangleF rectangle = new RectangleF(0, 0, 100, 100);
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Save();
//GrayScale color space.

```

```

graphics.ColorSpace = PdfColorSpace.GrayScale;
graphics.DrawRectangle(pen, brush, rectangle);
//CMYK color space.
graphics.ColorSpace = PdfColorSpace.CMYK;
graphics.DrawRectangle(pen, brush, rectangle);
graphics.Restore();
//Default color space.
graphics.DrawRectangle(pen, brush, rectangle);
//Draws by using the PdfBrush.
graphics.DrawRectangle(brush, rectangle);
//Save the document into stream.
MemoryStream stream = new MemoryStream();
loadedDocument.Save(stream);
//Close the document.
loadedDocument.Close(true);
//Save the stream into pdf file
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer PDF/Xamarin section for respective code
samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", stream);
}

```

Working with JavaScript

A JavaScript action allows execution of JavaScript code embedded in the PDF document. Essential PDF supports adding JavaScript action to the PDF document in the following:

- Document level JavaScript action
- JavaScript action to the form fields
- JavaScript in 3D Annotation

Document level JavaScript action

You can add the JavaScript action to the PDF document by using [PdfJavaScriptAction](#) class. The following code sample illustrates this.

C#

```

//Create a new document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create JavaScript action
PdfJavaScriptAction scriptAction = new
PdfJavaScriptAction("app.alert(\"Hello World!!!\")");
//Add the JavaScript action
document.Actions.AfterOpen = scriptAction;

```

```
//Save and close the PDF document
document.Save("Output.pdf");
document.Close(true);
```

VB.NET

```
'Create a new document
Dim document As New PdfDocument()
'Add a page
Dim page As PdfPage = document.Pages.Add()
'Create JavaScript action
Dim scriptAction As New PdfJavaScriptAction("app.alert(""Hello World!!!"")")
'Add the JavaScript action
document.Actions.AfterOpen = scriptAction
'Save and close the PDF document
document.Save("Output.pdf")
document.Close(True)
```

UWP

```
//Create a new document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create JavaScript action
PdfJavaScriptAction scriptAction = new
PdfJavaScriptAction("app.alert(\"Hello World!!!\")");
//Add the JavaScript action
document.Actions.AfterOpen = scriptAction;
MemoryStream memoryStream = new MemoryStream();
//Save the document
await document.SaveAsync(memoryStream);
//Close the documents
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(memoryStream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create JavaScript action
PdfJavaScriptAction scriptAction = new
PdfJavaScriptAction("app.alert(\"Hello World!!!\")");
//Add the JavaScript action
document.Actions.AfterOpen = scriptAction;
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document
document.Close(true);
```

```
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new document
PdfDocument document = new PdfDocument();
//Add a page
PdfPage page = document.Pages.Add();
//Create JavaScript action
PdfJavaScriptAction scriptAction = new
PdfJavaScriptAction("app.alert(\"Hello World!!!\")");
//Add the JavaScript action
document.Actions.AfterOpen = scriptAction;
//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
"application/pdf", memoryStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
"application/pdf", memoryStream);
}
```

JavaScript action to the form fields

You can add the JavaScript actions to various form fields using [PdfJavaScriptAction](#) instance. The [PdfFieldActions](#) class is used to create form field actions.

The following code snippet illustrate this.

C#

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Create a new PdfButtonField
PdfButtonField submitButton = new PdfButtonField(page, "submitButton");
submitButton.Bounds = new RectangleF(25, 160, 100, 20);
submitButton.Text = "Apply";
submitButton.BackColor = new PdfColor(181, 191, 203);
```

```
//Create a new PdfJavaScriptAction
PdfJavaScriptAction scriptAction = new PdfJavaScriptAction("app.alert(\"You
are looking at Form field action of PDF \")");
//Set the script action to submitButton
submitButton.Actions.MouseDown = scriptAction;
//Add the submit button to the new document
document.Form.Fields.Add(submitButton);
//Save document to disk
document.Save("fieldAction.pdf");
//Save document to disk
document.Close(true);
```

VB.NET

```
'Create a new PDF document
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Create a new PdfButtonField
Dim submitButton As New PdfButtonField(page, "submitButton")
submitButton.Bounds = New RectangleF(25, 160, 100, 20)
submitButton.Text = "Apply"
submitButton.BackColor = New PdfColor(181, 191, 203)
'Create a new PdfJavaScriptAction
Dim scriptAction As New PdfJavaScriptAction("app.alert(\"You are looking at
Form field action of PDF \")")
'Set the scriptAction to submitButton
submitButton.Actions.MouseDown = scriptAction
'Add the submit button to the new document
document.Form.Fields.Add(submitButton)
'Save document to disk
document.Save("fieldAction.pdf")
'Close the document
document.Close(True)
```

UWP

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Create a new PdfButtonField
PdfButtonField submitButton = new PdfButtonField(page, "submitButton");
submitButton.Bounds = new RectangleF(25, 160, 100, 20);
submitButton.Text = "Apply";
submitButton.BackColor = new PdfColor(181, 191, 203);
//Create a new PdfJavaScriptAction
PdfJavaScriptAction scriptAction = new PdfJavaScriptAction("app.alert(\"You
are looking at Form field action of PDF \")");
//Set the scriptAction to submitButton
submitButton.Actions.MouseDown = scriptAction;
//Add the submit button to the new document
document.Form.Fields.Add(submitButton);
MemoryStream memoryStream = new MemoryStream();
//Save the document
await document.SaveAsync(memoryStream);
```

```
//Close the document
document.Close(true);
//Save the stream as PDF document file in local machine. Refer to the
PDF/UWP section for respective code samples
Save(memoryStream, "Output.pdf");
```

ASP.NET CORE

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Create a new PdfButtonField
PdfButtonField submitButton = new PdfButtonField(page, "submitButton");
submitButton.Bounds = new RectangleF(25, 160, 100, 20);
submitButton.Text = "Apply";
submitButton.BackColor = new PdfColor(181, 191, 203);
//Create a new PdfJavaScriptAction
PdfJavaScriptAction scriptAction = new PdfJavaScriptAction("app.alert(\"You
are looking at Form field action of PDF \")");
//Set the scriptAction to submitButton
submitButton.Actions.MouseDown = scriptAction;
//Add the submit button to the new document
document.Form.Fields.Add(submitButton);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Close the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "Output.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);
```

XAMARIN

```
//Create a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Create a new PdfButtonField
PdfButtonField submitButton = new PdfButtonField(page, "submitButton");
submitButton.Bounds = new RectangleF(25, 160, 100, 20);
submitButton.Text = "Apply";
submitButton.BackColor = new PdfColor(181, 191, 203);
//Create a new PdfJavaScriptAction
PdfJavaScriptAction scriptAction = new PdfJavaScriptAction("app.alert(\"You
are looking at Form field action of PDF \")");
//Set the scriptAction to submitButton
submitButton.Actions.MouseDown = scriptAction;
//Add the submit button to the new document
document.Form.Fields.Add(submitButton);
```



```

//Save the document into stream
MemoryStream memoryStream = new MemoryStream();
document.Save(memoryStream);
//Close the documents
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pdf",
    "application/pdf", memoryStream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pdf",
    "application/pdf", memoryStream);
}

```

JavaScript in 3D Annotation

3D Annotations are used to represent 3D artworks in a PDF document. You can add a JavaScript code to 3D annotation using the [OnInstantiate](#) property in [Pdf3DAnnotation](#) instance. The JavaScript script is executed when the 3D artwork is instantiated. The following code snippet illustrate this.

C#

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
//Create a new PDF 3D annotation
Pdf3DAnnotation pdf3dAnnotation = new Pdf3DAnnotation(new RectangleF(10, 50,
300, 150), @"Input.u3d");
//Assign JavaScript script
pdf3dAnnotation.OnInstantiate = "host.getURL(\"http://www.google.com\")";
//Handles the activation of the 3D annotation
Pdf3DActivation activation = new Pdf3DActivation();
activation.ActivationMode = Pdf3DActivationMode.ExplicitActivation;
activation.ShowToolbar = true;
pdf3dAnnotation.Activation = activation;
//Adds annotation to page
page.Annotations.Add(pdf3dAnnotation);
//Save the document to disk
document.Save("Output.pdf");
//Close the document
document.Close(true);

```

VB.NET

```

'Creates a new PDF document
Dim document As New PdfDocument()
'Creates a new page
Dim page As PdfPage = document.Pages.Add()
'Create a new PDF 3D annotation

```

```

Dim pdf3dAnnotation As New Pdf3DAnnotation(New RectangleF(10, 50, 300, 150),
"Input.u3d")
'Assign JavaScript script
pdf3dAnnotation.OnInstantiate = "host.getURL(\"http://www.google.com\")"
'Handles the activation of the 3D annotation
Dim activation As New Pdf3DActivation()
activation.ActivationMode = Pdf3DActivationMode.ExplicitActivation
activation.ShowToolbar = True
pdf3dAnnotation.Activation = activation
'Adds annotation to page
page.Annotations.Add(pdf3dAnnotation)
'Save the document to disk
document.Save("Output.pdf")
'Close the document
document.Close(True)

```

UWP

```

//PDF supports 3D annotation only in Windows Forms, WPF, ASP.NET, ASP.NET
MVC, and ASP.NET Core platforms

```

ASP.NET CORE

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
FileStream inputStream = new FileStream("3DAnnotation.U3D", FileMode.Open,
FileAccess.Read);
//Creates a new PDF 3D annotation
Pdf3DAnnotation pdf3dAnnotation = new Pdf3DAnnotation(new RectangleF(10, 50,
300, 150), inputStream);
//Assign JavaScript script
pdf3dAnnotation.OnInstantiate = "host.getURL(\"http://www.google.com\")";
//Handles the activation of the 3D annotation
Pdf3DActivation activation = new Pdf3DActivation();
activation.ActivationMode = Pdf3DActivationMode.ExplicitActivation;
activation.ShowToolbar = true;
pdf3dAnnotation.Activation = activation;
//Adds annotation to page
page.Annotations.Add(pdf3dAnnotation);
//Save the document into stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
//Closes the document
document.Close(true);
//Defining the ContentType for PDF file
string contentType = "application/pdf";
//Define the file name
string fileName = "3DAnnotation.pdf";
//Creates a FileContentResult object by using the file contents, content
type, and file name
return File(stream, contentType, fileName);

```

XAMARIN

```

//Creates a new PDF document
PdfDocument document = new PdfDocument();
//Creates a new page
PdfPage page = document.Pages.Add();
Stream inputStream =
typeof(MainPage).GetTypeInfo().Assembly.GetManifestResourceStream("Sample.As
sets.3DAnnotation.u3d");
//Creates a new PDF 3d annotation
Pdf3DAnnotation pdf3dAnnotation = new Pdf3DAnnotation(new RectangleF(10, 50,
300, 150), inputStream);
//Assign JavaScript script
pdf3dAnnotation.OnInstantiate = "host.getURL(\"http://www.google.com\")";
//Adds annotation to page
page.Annotations.Add(pdf3dAnnotation);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Close the document
document.Close(true);
//Save the stream into PDF file
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the PDF/Xamarin section for respective code
samples
if (Device.RuntimePlatform == Device.UWP)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("3DAnnotation.
pdf", "application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("3DAnnotation.pdf",
"application/pdf", stream);
}

```

Supported and Unsupported Features

The following table shows the various features available in the Essential PDF and their availability in different platforms.

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
Drawing Text	Yes	Yes	Yes	Yes	Yes
Text Formatting	Yes	Yes	Yes	Yes	Yes
Multilingual Support	Yes	No	No	No	No
Drawing RTL text	Yes	Yes	Yes	Yes	Yes

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
Drawing complex script text	Yes	Yes	Yes	Yes	Yes
Text Extraction	Yes	Yes	No	No	No
Unicode	Yes	Yes	Yes	Yes	Yes
Pagination	Yes	Yes	Yes	Yes	Yes

Graphics:

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
Pen and Brush	Yes	Yes	Yes	Yes	Yes
Layers	Yes	Yes	Yes	Yes	Yes
Transparent Graphics	Yes	No	No	No	No
Color Spaces	Yes	Yes	Yes	Yes	Yes
Image Extraction	Yes	No	No	No	No
Enhanced Printing Support	Yes	Yes	Yes	Yes	Yes
Barcode	Yes	Yes	Yes	Yes	Yes

Document-level Operations :

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
Merge Documents	Yes	Yes	Yes	Yes	Yes
Split Document	Yes	Yes	Yes	Yes	Yes

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
Overlay Documents	Yes	Yes	Yes	Yes	Yes
Import Pages	Yes	Yes	Yes	Yes	Yes
Stamp	Yes	Yes	Yes	Yes	Yes
Booklet	Yes	Yes	Yes	Yes	Yes

Document Settings :

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
Custom Metadata	Yes	Yes	Yes	Yes	Yes
Document Properties	Yes	Yes	Yes	Yes	Yes
Page Orientation	Yes	Yes	Yes	Yes	Yes
Page Sizes	Yes	Yes	Yes	Yes	Yes
Viewer Preferences	Yes	Yes	Yes	Yes	Yes
Tagged PDF with section 508 compliant	Yes	Yes	Yes	Yes	Yes

Forms:

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
Fields	Yes	Yes	Yes	Yes	Yes
Form Filling	Yes	Yes	Yes	Yes	Yes

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
Flatten	Yes	Yes	Yes	Yes	Yes
Import Form Data	Yes	Yes	Yes	Yes	Yes
Form Export	Yes	Yes	Yes	Yes	Yes

XFA Forms:

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
Fields	Yes	Yes	Yes*	Yes	Yes
Form Filling	Yes	Yes	Yes*	Yes	Yes

*Supported on ASP.NET Core 2.0 and above

Document Conversion:

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
TIFF to PDF	Yes	Yes	Yes*	No	No
HTML to PDF	Yes	No	Yes*	No	Yes
RTF To PDF	Yes	Yes	Yes	No	Yes
DOC To PDF	Yes	Yes	Yes	No	Yes
Excel To PDF	Yes	Yes*	Yes*	No	Yes
PDF OCR	Yes	No	No	No	No
XPS to PDF	Yes	Yes	No	No	No
SVG to PDF	Yes	No	No	No	No
EMF to PDF	Yes	No	No	No	No

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
PDF to PDF/A-1b	Yes	No	No	No	No

*Supported on .NETStandard 2.0 and above

PDF Standards:

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
PDF/ A-1b Compliance	Yes	Yes	Yes	Yes	Yes
PDF/ A-2b Compliance	Yes	Yes	Yes	Yes	Yes
PDF/ A-3b Compliance	Yes	Yes	Yes	Yes	Yes
PDF/x1a: 2001 Compliance	Yes	Yes	No	No	No
ZUGFeRD Invoice	Yes	Yes	Yes	Yes	Yes

Fonts:

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
Standard Fonts	Yes	Yes	Yes	Yes	Yes
CJK Fonts	Yes	Yes	Yes	Yes	Yes
TrueType Fonts	Yes	Yes	Yes	Yes	Yes
Unicode TrueType	Yes	Yes	Yes	Yes	Yes

Images:

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
Scalar Images	Yes	Yes	Yes	Yes	Yes
Soft Mask	Yes	Yes	Yes	Yes	Yes
Vector Images	Yes	No	No	No	No
Watermarks	Yes	Yes	Yes	Yes	Yes

Tables:

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
ADO.Net Tables Support	Yes	No	No	No	No
Cell / Row / Column Formatting	Yes	Yes	Yes	Yes	Yes
Header	Yes	Yes	Yes	Yes	Yes
Pagination	Yes	Yes	Yes	Yes	Yes
Borders	Yes	Yes	Yes	Yes	Yes
RowSpan and ColumnSpan	Yes	Yes	Yes	Yes	Yes
Nested table	Yes	Yes	Yes	Yes	Yes
Cell Padding and Spacing	Yes	Yes	Yes	Yes	Yes

Page Level Operations:

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
Headers and Footers	Yes	Yes	Yes	Yes	Yes
Page Label	Yes	Yes	Yes	Yes	Yes
Automatic Fields	Yes	Yes	Yes	Yes	Yes

Interactive Elements:

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
3D-Annotation	Yes	Yes	Yes	Yes	Yes
Measurement Annotations	Yes	Yes	Yes	Yes	Yes
Action	Yes	Yes	Yes	Yes	Yes
Attachment	Yes	Yes	Yes	Yes	Yes
Bookmark	Yes	Yes	Yes	Yes	Yes
Hyperlink	Yes	Yes	Yes	Yes	Yes
Portfolio	Yes	Yes	Yes	Yes	Yes
Import Annotation Data	Yes	Yes	Yes	Yes	Yes
Export Annotation Data	Yes	Yes	Yes	Yes	Yes

Security:

Features	.NET Framework	UWP	Xamarin and .NET Core	Blazor Client-Side	Blazor Server-Side & Hosted Application
Digital Signature	Yes	Yes	Yes	Yes	Yes
Digital Signature with LTV	Yes	Yes	Yes	Yes	Yes
Validate Digital Signature	Yes	Yes	Yes	Yes	Yes
Encryption and Decryption	Yes	Yes	Yes	Yes	Yes
Redaction	Yes	No	No	No	No

Presentation

Overview of PowerPoint Presentation

Essential Presentation is a native **.NET** class library that can be used by developers to create, read, and write Microsoft PowerPoint files by using C#, VB.NET, and managed C++ code. The library can be used in Windows Forms, WPF, ASP.NET, ASP.NET MVC, UWP and Xamarin platforms.

It is a non-UI component that provides a full-fledged PowerPoint presentation instance that facilitates accessing and manipulating the presentations without any dependency of Microsoft Office COM libraries and Microsoft Office.

Key features

- Support to [create](#) PowerPoint presentation from scratch.
- [Open](#), [modify](#), and [save](#) existing presentations.
- Ability to [convert PowerPoint presentation to PDF](#).
- Ability to [convert PowerPoint slides to images](#).
- Ability to [create](#) and [edit](#) charts.
- Ability to [convert chart in a slide to image](#).
- Ability to [clone](#) and [merge](#) slides in presentation
- Ability to [create](#) and [edit](#) animations.
- Ability to [create](#) and [edit](#) transition effects.
- Ability to [create](#) and [edit](#) comments.
- Ability to [encrypt](#) and [decrypt](#) PowerPoint presentation.
- Ability to [set and remove write protection](#) of PowerPoint presentation.
- Ability to access the [Built-in](#) and [Custom](#) document properties.
- Ability to [create](#) and [modify](#) sections in PowerPoint presentation.

Compatible Microsoft PowerPoint Versions

- Microsoft PowerPoint 2007
- Microsoft PowerPoint 2010
- Microsoft PowerPoint 2013
- Microsoft PowerPoint 2016
- Microsoft PowerPoint 2019

Note: 1. The current version of Essential Presentation supports the .PPTX, .PPTM, .POTX, .POTM file formats only.

2. The current version of Essential Presentation does not support some features in Microsoft PowerPoint such as Word Art, creation and editing of Handouts, equations, create and edit audio and video content, built-in themes, and its variants.

Assemblies Required

The following assemblies need to be referenced in your application

Platform(s)	Assembly
WPF, Windows Forms, ASP. NET and ASP.NET MVC	Syncfusion.Presentation.Base Syncfusion.Compression.Base Syncfusion.OfficeChart.Base
ASP.NET Core, Xamarin and Blazor	Syncfusion.Presentation.Portable Syncfusion.Compression.Portable Syncfusion.OfficeChart.Portable
Universal Windows Platform	Syncfusion.Presentation.UWP Syncfusion.OfficeChart.UWP

Note: Starting with v16.2.0.x, if you reference Syncfusion assemblies from trial setup or from the NuGet feed, you also have to include a license key in your projects. Please refer to this [link](#) to know about registering Syncfusion license key in your applications to use our components.

Converting PowerPoint Presentation to PDF

For converting a PowerPoint Presentation to PDF, the following assemblies needed to be referenced in your application

Platform(s)	Assembly
WPF, Windows Forms, ASP. NET and ASP.NET MVC	Syncfusion.Presentation.Base Syncfusion.Compression.Base Syncfusion.OfficeChart.Base Syncfusion.Pdf.Base Syncfusion.PresentationToPDFConverter.Base
ASP.NET Core, Xamarin and Blazor	Syncfusion.Presentation.Portable Syncfusion.Compression.Portable Syncfusion.OfficeChart.Portable Syncfusion.Pdf.Portable

	Syncfusion.PresentationRenderer.Portable Syncfusion.SkiaSharpHelper.Portable Skiasharp
--	--

The following assemblies are required to be referred in addition to the above mentioned assemblies for converting the chart present in the PowerPoint Presentation into PDF.

Platform(s)	Assembly
WPF, Windows Forms, ASP. NET and ASP.NET MVC	Syncfusion.OfficeChartToImageConverter.WPF Syncfusion.SfChart.WPF

Note: 1.The “Syncfusion.OfficeChartToImageConverter.WPF” assembly is supported from .NET Framework 4.0 onwards

NuGet Packages Required

To work with PowerPoint Presentations, install the following NuGet packages in your application:

Platform(s)	NuGet Package
Windows Forms, Console Application (Targeting .NET Framework)	Syncfusion.Presentation.WinForms.nupkg
WPF	Syncfusion.Presentation.Wpf.nupkg
.NET Framework 3.5 or 4.0 Client Profile	Syncfusion.Presentation.ClientProfile.nupkg
ASP.NET Web Forms	Syncfusion.Presentation.AspNet.nupkg
ASP.NET MVC4	Syncfusion.Presentation.AspNet.Mvc4.nupkg
ASP.NET MVC5	Syncfusion.Presentation.AspNet.Mvc5.nupkg
UWP	Syncfusion.Presentation.UWP.nupkg
ASP.NET Core, Console Application (Targeting .NET Core) and Blazor	Syncfusion.Presentation.Net.Core.nupkg
Xamarin	Syncfusion.Xamarin.Presentation.nupkg

Note: 1.Starting with v16.2.0.x, if you reference Syncfusion assemblies from trial setup or from the NuGet feed, add the "Syncfusion.Licensing" assembly reference and include a license key in your projects. Refer to this [link](#) to learn about registering Syncfusion license key in your applications to use the components.

2.From the Essential Studio 2018 Volume 3 release(v16.3.0.21), Syncfusion has changed some of the NuGet package names to search and find the required Syncfusion NuGet packages in nuget.org easily based on the control and its platforms.

Converting PowerPoint Presentation into PDF

For converting PowerPoint Presentation into PDF, install the following NuGet packages in your application:

Platform(s)	NuGet Package
Windows Forms, Console Application (Targeting .NET Framework)	Syncfusion.PresentationToPDFConverter.WinForms.nupkg
WPF	Syncfusion.PresentationToPDFConverter.Wpf.nupkg
.NET Framework 3.5 or 4.0 Client Profile	Syncfusion.PresentationToPDFConverter.ClientProfile.nupkg
ASP.NET Web	Syncfusion.PresentationToPDFConverter.AspNet.nupkg
ASP.NET MVC4	Syncfusion.PresentationToPDFConverter.AspNet.Mvc4.nupkg
ASP.NET MVC5	Syncfusion.PresentationToPDFConverter.AspNet.Mvc5.nupkg
UWP, ASP.NET Core, Console Application (Targeting .NET Core) and Blazor	Syncfusion.PresentationRenderer.Net.Core.nupkg
	Syncfusion.SkiaSharpHelper.Net.Core.nupkg
Xamarin	Syncfusion.Xamarin.PresentationRenderer.nupkg
	Syncfusion.Xamarin.SkiaSharpHelper.nupkg

Note: PowerPoint Presentation to PDF conversion is supported from .NET Standard 1.4 onwards for ASP.NET Core and Xamarin.

Converting PowerPoint Presentation to image

For converting a PowerPoint Presentation to image, install the following NuGet packages in your application:

Platform(s)	NuGet Package
Windows Forms, Console Application (Targeting .NET Framework)	Syncfusion.Presentation.WinForms.nupkg
WPF	Syncfusion.Presentation.Wpf.nupkg
.NET Framework 3.5 or 4.0 Client Profile	Syncfusion.Presentation.ClientProfile.nupkg
ASP.NET Web Forms	Syncfusion.Presentation.AspNet.nupkg
ASP.NET MVC4	Syncfusion.Presentation.AspNet.Mvc4.nupkg
ASP.NET MVC5	Syncfusion.Presentation.AspNet.Mvc5.nupkg
UWP	Syncfusion.Presentation.UWP.nupkg
ASP.NET Core, Console Application (Targeting .NET Core) and Blazor	Syncfusion.PresentationRenderer.Net.Core.nupkg
	Syncfusion.SkiaSharpHelper.Net.Core.nupkg

Xamarin	Syncfusion.Xamarin.PresentationRenderer.nupkg
	Syncfusion.Xamarin.SkiaSharpHelper.nupkg

Note: PowerPoint Presentation to image conversion is supported from .NET Framework 2.0 and .NET Standard 1.4 onwards for ASP.NET Core and Xamarin.

Converting charts in Presentation

The following NuGet package should be installed additionally to convert the charts present in PowerPoint Presentation:

Platform(s)	NuGet Package
Windows Forms, Console Application (Targeting .NET Framework)	Syncfusion.OfficeChartToImageConverter.WinForms.nupkg
WPF	Syncfusion.OfficeChartToImageConverter.Wpf.nupkg
ASP.NET Web	Syncfusion.OfficeChartToImageConverter.AspNet.nupkg
ASP.NET MVC4	Syncfusion.OfficeChartToImageConverter.AspNet.Mvc4.nupkg
ASP.NET MVC5	Syncfusion.OfficeChartToImageConverter.AspNet.Mvc5.nupkg
UWP	Syncfusion.OfficeChartToImageConverter.UWP.nupkg
ASP.NET Core, Console Application (Targeting .NET Core) and Blazor	Syncfusion.PresentationRenderer.Net.Core.nupkg
	Syncfusion.SkiaSharpHelper.Net.Core.nupkg
Xamarin	Syncfusion.Xamarin.PresentationRenderer.nupkg
	Syncfusion.Xamarin.SkiaSharpHelper.nupkg

Note: 1. The "Syncfusion.OfficeChartToImageConverter.Wpf.nupkg" NuGet package is supported only from 4.0 .NET Framework onwards.

2. The "Syncfusion.Xamarin.PresentationRenderer.nupkg" and "Syncfusion.PresentationRenderer.Net.Core.nupkg" NuGet packages supports chart to image conversion only from .NET Standard 2.0 onwards.

NuGet package installation and uninstallation

To use NuGet package in your project, refer to the NuGet package [Installation](#) and [Uninstallation](#) sections.

The PowerPoint Presentation NuGet packages can be installed and uninstalled using the Package Manager Console. In Visual Studio, select **Tools > NuGet Package Manager > Package Manager Console** and execute the following commands in respective platforms.

Note: The Syncfusion components are available in nuget.org

Platform(s)	Install	Uninstall
Windows Forms	<i>Install-package</i> <i>Syncfusion.Presentation.WinForms</i> Install-package Syncfusion.PresentationToPdfConverter.WinForms * Install-package Syncfusion.OfficeChartToImageConverter.WinForms	<i>Uninstall-package</i> <i>Syncfusion.Presentation.WinForms - RemoveDependencies</i> Uninstall-package Syncfusion.PresentationToPdfConverter.WinForms -RemoveDependencies * Uninstall-package Syncfusion.OfficeChartToImageConverter.WinForms -RemoveDependencies
WPF	<i>Install-package Syncfusion.Presentation.Wpf</i> Install-package Syncfusion.PresentationToPdfConverter.Wpf * Install-package Syncfusion.OfficeChartToImageConverter.Wpf	<i>Uninstall-package</i> <i>Syncfusion.Presentation.Wpf - RemoveDependencies</i> Uninstall-package Syncfusion.PresentationToPdfConverter.Wpf - RemoveDependencies * Uninstall-package Syncfusion.OfficeChartToImageConverter.Wpf -RemoveDependencies
ASP.NET Web	<i>Install-package</i> <i>Syncfusion.Presentation.AspNet</i> Install-package Syncfusion.PresentationToPdfConverter.AspNet * Install-package Syncfusion.OfficeChartToImageConverter.AspNet	<i>Uninstall-package</i> <i>Syncfusion.Presentation.AspNet - RemoveDependencies</i> Uninstall-package Syncfusion.PresentationToPdfConverter.AspNet -RemoveDependencies * Uninstall-package Syncfusion.OfficeChartToImageConverter.AspNet -RemoveDependencies
ASP.NET MVC4	<i>Install-package</i> <i>Syncfusion.Presentation.AspNet.MVC4</i> Install-package Syncfusion.PresentationToPdfConverter.AspNet.MVC4 * Install-package Syncfusion.OfficeChartToImageConverter.AspNet.MVC4	<i>Uninstall-package</i> <i>Syncfusion.Presentation.AspNet.MVC4 - RemoveDependencies</i> Uninstall-package Syncfusion.PresentationToPdfConverter.AspNet.MVC4 -RemoveDependencies * Uninstall-package

		Syncfusion.OfficeChartToImageConverter.AspNet.MVC4 -RemoveDependencies
ASP.NET MVC5	<i>Install-package</i> Syncfusion.Presentation.AspNet.MVC5 Install-package Syncfusion.PresentationToPdfConverter.AspNet.MVC5 * Install-package Syncfusion.OfficeChartToImageConverter.AspNet.MVC5	<i>Uninstall-package</i> Syncfusion.Presentation.AspNet.MVC5 - RemoveDependencies Uninstall-package Syncfusion.PresentationToPdfConverter.AspNet.MVC5 -RemoveDependencies * Uninstall-package Syncfusion.OfficeChartToImageConverter.AspNet.MVC5 -RemoveDependencies
UWP	<i>Install-package</i> Syncfusion.Presentation.UWP Install-package Syncfusion.OfficeChartToImageConverter.UWP	<i>Uninstall-package</i> Syncfusion.Presentation.UWP â€”RemoveDependencies Uninstall-package Syncfusion.OfficeChartToImageConverter.UWP -RemoveDependencies
ASP.NET Core and Blazor	<i>Install-package</i> Syncfusion.Presentation.Net.Core Install-package Syncfusion.PresentationRenderer.Net.Core * Install-package Syncfusion.SkiaSharpHelper.Net.Core	<i>Uninstall-package</i> Syncfusion.Presentation.Net.Core â€”RemoveDependencies Uninstall-package Syncfusion.PresentationRenderer.Net.Core - RemoveDependencies * Uninstall-package Syncfusion.SkiaSharpHelper.Net.Core â€” RemoveDependencies
Xamarin	<i>Install-package</i> Syncfusion.Xamarin.Presentation Install-package Syncfusion.Xamarin.PresentationRenderer * Install-package Syncfusion.Xamarin.SkiaSharpHelper	<i>Uninstall-package</i> Syncfusion.Xamarin.Presentation â€”RemoveDependencies Uninstall-package Syncfusion.Xamarin.PresentationRenderer - RemoveDependencies * Uninstall-package Syncfusion.Xamarin.SkiaSharpHelper - RemoveDependencies

Getting Started

Creating a simple PowerPoint Presentation with basic elements from scratch

In this page, you can learn how to create a simple PowerPoint Presentation by using Essential Presentation API.

For creating and manipulating a PowerPoint Presentation, include the following assemblies in the application.

Assembly Name	Short Description
Syncfusion.Presentation.Base	This assembly contains the core features required for creating, reading, manipulating a Presentation file.
Syncfusion.Compression.Base	This assembly is used to package the Presentation contents.
Syncfusion.OfficeChart.Base	This assembly contains the office chart object model and core features needed for chart creation.

Note: Starting with v16.2.0.x, if you reference Syncfusion assemblies from trial setup or from the NuGet feed, you also have to include a license key in your projects. Please refer to this [link](#) to know about registering Syncfusion license key in your applications to use our components.

Include the following namespace in your .cs or .vb code as shown below

C#

```
using Syncfusion.Presentation;
```

VB.NET

```
Imports Syncfusion.Presentation
```

UWP

```
using Syncfusion.Presentation;
```

ASP.NET CORE

```
using Syncfusion.Presentation;
```

XAMARIN

```
using Syncfusion.Presentation;
```

An entire PowerPoint Presentation is represented by an instance of 'IPresentation' interface and it is the root element of Essential Presentation's DOM.

The following code example demonstrates how to create an instance of 'IPresentation' interface.

C#

```
//Creates a new instance of PowerPoint presentation
```

```
IPresentation pptxDoc = Presentation.Create();
```

VB.NET

```
'Creates a new instance of PowerPoint presentation  
Dim pptxDoc As IPresentation = Presentation.Create()
```

UWP

```
//Creates a new instance of PowerPoint presentation  
IPresentation pptxDoc = Presentation.Create();
```

ASP.NET CORE

```
//Creates a new instance of PowerPoint presentation  
IPresentation pptxDoc = Presentation.Create();
```

XAMARIN

```
//Creates a new instance of PowerPoint presentation  
IPresentation pptxDoc = Presentation.Create();
```

'IPresentation' instance has a slide collection that represents the individual slides present within PowerPoint presentation. A slide may contain textual and other graphics contents like shapes, images, charts etc.

The following code example demonstrates how to add a blank slide to a PowerPoint Presentation.

C#

```
//Adds a slide to the PowerPoint Presentation  
ISlide firstSlide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
```

VB.NET

```
'Adds a slide to the PowerPoint Presentation  
Dim firstSlide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
```

UWP

```
//Adds a slide to the PowerPoint Presentation  
ISlide firstSlide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
```

ASP.NET CORE

```
//Adds a slide to the PowerPoint Presentation  
ISlide firstSlide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
```

XAMARIN

```
//Adds a slide to the PowerPoint Presentation  
ISlide firstSlide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
```

Note: The 'Point' typographic units are used to add or manipulate any element in a Presentation.

All the textual contents in a Presentation document are represented by paragraphs. Within the paragraph, textual contents are grouped into one or more child elements as 'TextParts'. Each 'TextPart' represents a region of text with a common set of formatted text.

The following code example demonstrates how to add text into a presentation.

C#

```
//Adds a textbox in a slide by specifying its position and size
IShape textShape = firstSlide.AddTextBox(100, 75, 756, 200);
//Adds a paragraph into the textShape
IParagraph paragraph = textShape.TextBody.AddParagraph();
//Set the horizontal alignment of paragraph
paragraph.HorizontalAlignment = HorizontalAlignmentType.Center;
//Adds a textPart in the paragraph
ITextPart textPart = paragraph.AddTextPart("Hello Presentation");
//Applies font formatting to the text
textPart.Font.FontSize = 80;
textPart.Font.Bold = true;
```

VB.NET

```
'Adds a textbox in a slide by specifying its position and size
Dim textShape As IShape = firstSlide.AddTextBox(100, 75, 756, 200)
'Adds a paragraph into the textShape
Dim paragraph As IParagraph = textShape.TextBody.AddParagraph()
'Set the horizontal alignment of paragraph
paragraph.HorizontalAlignment = HorizontalAlignmentType.Center
'Add a textPart in the paragraph
Dim textPart As ITextPart = paragraph.AddTextPart("Hello Presentation")
'Applies font formatting to the text
textPart.Font.FontSize = 80
textPart.Font.Bold = True
```

UWP

```
//Adds a textbox in a slide by specifying its position and size
IShape textShape = firstSlide.AddTextBox(100, 75, 756, 200);
//Adds a paragraph into the textShape
IParagraph paragraph = textShape.TextBody.AddParagraph();
//Set the horizontal alignment of paragraph
paragraph.HorizontalAlignment = HorizontalAlignmentType.Center;
//Adds a textPart in the paragraph
ITextPart textPart = paragraph.AddTextPart("Hello Presentation");
//Applies font formatting to the text
textPart.Font.FontSize = 80;
textPart.Font.Bold = true;
```

ASP.NET CORE

```
//Adds a textbox in a slide by specifying its position and size
IShape textShape = firstSlide.AddTextBox(100, 75, 756, 200);
```

```
//Adds a paragraph into the textShape
IParagraph paragraph = textShape.TextBody.AddParagraph();
//Set the horizontal alignment of paragraph
paragraph.HorizontalAlignment = HorizontalAlignmentType.Center;
//Adds a textPart in the paragraph
ITextPart textPart = paragraph.AddTextPart("Hello Presentation");
//Applies font formatting to the text
textPart.Font.FontSize = 80;
textPart.Font.Bold = true;
```

XAMARIN

```
//Adds a textbox in a slide by specifying its position and size
IShape textShape = firstSlide.AddTextBox(100, 75, 756, 200);
//Adds a paragraph into the textShape
IParagraph paragraph = textShape.TextBody.AddParagraph();
//Set the horizontal alignment of paragraph
paragraph.HorizontalAlignment = HorizontalAlignmentType.Center;
//Adds a textPart in the paragraph
ITextPart textPart = paragraph.AddTextPart("Hello Presentation");
//Applies font formatting to the text
textPart.Font.FontSize = 80;
textPart.Font.Bold = true;
```

Essential Presentation allows you to create simple and multi-level lists that make the content easier for reading. The following code example demonstrates how to add a bulleted list in a paragraph.

C#

```
//Adds a new paragraph with text.
paragraph = textShape.TextBody.AddParagraph("AdventureWorks Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company.");
//Sets the list type as bullet
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);
//Sets the font of the bullet character
paragraph.ListFormat.FontName = "Symbol";
//Sets the hanging value as 20
paragraph.FirstLineIndent = -20;
```

VB.NET

```
'Adds a new paragraph with text.
paragraph = textShape.TextBody.AddParagraph("AdventureWorks Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company.")
'Sets the list type as bullet
paragraph.ListFormat.Type = ListType.Bulleted
'Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183)
'Sets the font of the bullet character
paragraph.ListFormat.FontName = "Symbol"
'Sets the hanging value as 20
```

```
paragraph.FirstLineIndent = -20
```

UWP

```
//Adds a new paragraph with text.
paragraph = textShape.TextBody.AddParagraph("AdventureWorks Cycles, the
fictitious company on which the AdventureWorks sample databases are based,
is a large, multinational manufacturing company.");
//Sets the list type as bullet
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);
//Sets the font of the bullet character
paragraph.ListFormat.FontName = "Symbol";
//Sets the hanging value as 20
paragraph.FirstLineIndent = -20;
```

ASP.NET CORE

```
//Adds a new paragraph with text.
paragraph = textShape.TextBody.AddParagraph("AdventureWorks Cycles, the
fictitious company on which the AdventureWorks sample databases are based,
is a large, multinational manufacturing company.");
//Sets the list type as bullet
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);
//Sets the font of the bullet character
paragraph.ListFormat.FontName = "Symbol";
//Sets the hanging value as 20
paragraph.FirstLineIndent = -20;
```

XAMARIN

```
//Adds a new paragraph with text.
paragraph = textShape.TextBody.AddParagraph("AdventureWorks Cycles, the
fictitious company on which the AdventureWorks sample databases are based,
is a large, multinational manufacturing company.");
//Sets the list type as bullet
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);
//Sets the font of the bullet character
paragraph.ListFormat.FontName = "Symbol";
//Sets the hanging value as 20
paragraph.FirstLineIndent = -20;
```

In PowerPoint Presentation, the multilevel lists are used for presenting the content in a hierarchy. You can create a multi-level list by setting the indentation levels. By default, the level begins at 0 and increments by 1 for each level. The following code example demonstrates how to add multi-level list in a paragraph.

C#

```
//Adds a new paragraph
paragraph = textShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as bullet
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the list level as 2. Possible values can range from 0 to 8
paragraph.IndentLevelNumber = 2;
```

VB.NET

```
'Adds a new paragraph
paragraph = textShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.")
'Sets the list type as bullet
paragraph.ListFormat.Type = ListType.Bulleted
'Sets the list level as 2. Possible values can range from 0 to 8
paragraph.IndentLevelNumber = 2
```

UWP

```
//Adds a new paragraph
paragraph = textShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as bullet
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the list level as 2. Possible values can range from 0 to 8
paragraph.IndentLevelNumber = 2;
```

ASP.NET CORE

```
//Adds a new paragraph
paragraph = textShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as bullet
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the list level as 2. Possible values can range from 0 to 8
paragraph.IndentLevelNumber = 2;
```

XAMARIN

```
//Adds a new paragraph
paragraph = textShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as bullet
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the list level as 2. Possible values can range from 0 to 8
paragraph.IndentLevelNumber = 2;
```

You can add images to the Presentation by adding them in the picture collection of a slide. The following code example demonstrates how to add an image in a presentation.

C#

```
//Gets the image from file path
Image image = Image.FromFile(@"image.jpg");
// Adds the image to the slide by specifying position and size
firstSlide.Pictures.AddPicture(new MemoryStream(image.ImageData), 300, 270,
410, 250);
```

VB.NET

```
'Gets the image from file path
Dim image__1 As Image = Image.FromFile("image.jpg")
' Adds the image to the slide by specifying position and size
firstSlide.Pictures.AddPicture(New MemoryStream (image__1.ImageData), 300,
270, 410, 250)
```

UWP

```
//Gets the image from file path
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream imageStream =
assembly.GetManifestResourceStream("UWP.Data.tablet.jpg");
// Adds the image to the slide by specifying position and size
firstSlide.Pictures.AddPicture(imageStream, 300, 270, 410, 250);
```

ASP.NET CORE

```
//Gets the image from file path
FileStream imageStream = new FileStream(@"Image.png", FileMode.Open,
FileAccess.Read);
// Adds the image to the slide by specifying position and size
firstSlide.Pictures.AddPicture(imageStream, 300, 270, 410, 250);
```

XAMARIN

```
//Gets the image from file path
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream imageStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.tablet.jpg");
// Adds the image to the slide by specifying position and size
firstSlide.Pictures.AddPicture(imageStream, 300, 270, 410, 250);
```

Finally, save the Presentation in file system and close its instance.

C#

```
//Saves the Presentation in the given name
pptxDoc.Save("Output.pptx");
//Releases the resources occupied
pptxDoc.Close();
```

VB.NET

```
'Saves the Presentation in the given name
pptxDoc.Save("Output.pptx")
'Releases the resources occupied
pptxDoc.Close()
```

UWP

```
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
//Releases the resources occupied
pptxDoc.Close();
```

ASP.NET CORE

```
//Saving the PowerPoint Presentation as stream
FileStream stream = new FileStream("Sample.pptx", FileMode.Create,
FileAccess.ReadWrite);
pptxDoc.Save(stream);
//Dispose stream
stream.Dispose();
//Close the presentation
pptxDoc.Close();
```

XAMARIN

```
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

The resultant PowerPoint Presentation looks as follows.

Hello Presentation

- Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company.
 - The company manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets.



Converting PowerPoint Presentation to PDF

Essential Presentation allows you to convert a PowerPoint Presentation into PDF document. The following assemblies are required for the Presentation to PDF conversion.

Assembly Name	Short Description
Syncfusion.Presentation.Base	This assembly contains the core features required for creating, reading, manipulating a Presentation file.
Syncfusion.Compression.Base	This assembly is used to pack the Presentation contents.
Syncfusion.OfficeChart.Base	This assembly contains the office chart object model and core features needed for chart creation.
Syncfusion.OfficeChartToImageConverter.WPF	This assembly is used to convert Office Chart into Image.
Syncfusion.Pdf.Base	This assembly is used for PDF file creation.
Syncfusion.PresentationToPDFConverter.Base	This assembly is used to convert Presentation file into PDF.
Syncfusion.SfChart.WPF	Supporting assembly for Syncfusion.OfficeChartToImageConverter.WPF

Include the following namespaces in your .cs or .vb code as shown below

C#

```
using Syncfusion.Presentation;
```

```
using Syncfusion.OfficeChartToImageConverter;
using Syncfusion.Pdf;
using Syncfusion.PresentationToPdfConverter;
```

VB.NET

```
Imports Syncfusion.Presentation
Imports Syncfusion.OfficeChartToImageConverter
Imports Syncfusion.Pdf
Imports Syncfusion.PresentationToPdfConverter
```

PresentationToPdfConverter class is responsible for converting an entire Presentation or a slide into PDF. The following code example demonstrates how to convert the PowerPoint presentation to PDF.

C#

```
//Opens a PowerPoint Presentation file
IPresentation pptxDoc = Presentation.Open(fileName);
//Creates an instance of ChartToImageConverter and assigns it to
ChartToImageConverter property of Presentation
pptxDoc.ChartToImageConverter = new ChartToImageConverter();
//Converts the PowerPoint Presentation into PDF document
PdfDocument pdfDocument = PresentationToPdfConverter.Convert(pptxDoc);
//Saves the PDF document
pdfDocument.Save(@"SampleWithoutSetting.pdf");
//Closes the PDF document
pdfDocument.Close(true);
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Opens a PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open(fileName)
'Creates an instance of ChartToImageConverter and assigns it to
ChartToImageConverter property of Presentation
pptxDoc.ChartToImageConverter = New ChartToImageConverter ()
'Converts the PowerPoint Presentation into PDF document
Dim pdfDocument As PdfDocument = PresentationToPdfConverter.Convert(pptxDoc)
'Saves the PDF document
pdfDocument.Save("SampleWithoutSetting.pdf")
'Closes the PDF document
pdfDocument.Close(True)
'Closes the Presentation
pptxDoc.Close()
```

UWP

```
//Essential Presentation Library does not support presentation to Pdf
conversion in UWP platform.
```

ASP.NET CORE

```
//Open the existing PowerPoint presentation.
```

```

string basePath = _hostingEnvironment.WebRootPath;
FileStream fileStreamInput = new FileStream(basePath +
@"/Presentation/ConversionTemplate.pptx", FileMode.Open, FileAccess.Read);
IPresentation pptxDoc = Presentation.Open(fileStreamInput);
//Convert the PowerPoint document to PDF document.
PdfDocument pdfDocument = PresentationToPdfConverter.Convert(pptxDoc);
//Save the converted PDF document to Memory stream.
MemoryStream pdfStream = new MemoryStream();
pdfDocument.Save(pdfStream);
pdfStream.Position = 0;
//Close the PDF document.
pdfDocument.Close(true);
//Close the PowerPoint Presentation.
pptxDoc.Close();
//Initialize the file stream to download the converted PDF.
FileStreamResult fileStreamResult = new FileStreamResult(pdfStream,
"application/pdf");
//Set the file name.
fileStreamResult.FileNameDownloadName = "Sample.pdf";
return fileStreamResult;

```

XAMARIN

```

string resourcePath =
"SampleBrowser.Presentation.Samples.Templates.Template.pptx";
Assembly assembly = typeof(GettingStarted).GetTypeInfo().Assembly;
Stream fileStream = assembly.GetManifestResourceStream(resourcePath);
//Open a PowerPoint presentation
IPresentation pptxDoc = Presentation.Open(fileStream);
//Convert the PowerPoint document to PDF document.
PdfDocument pdfDocument = PresentationToPdfConverter.Convert(pptxDoc);
//Save the converted PDF document.
MemoryStream pdfStream = new MemoryStream();
pdfDocument.Save(pdfStream);
pdfStream.Position = 0;
//Close the PDF document.
pdfDocument.Close(true);
//Close the PowerPoint Presentation.
pptxDoc.Close();
if (Device.RuntimePlatform == Device.UWP)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("PPTXToPDF.pdf",
"application/pdf", pdfStream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("PPTXToPDF.pdf",
"application/pdf", pdfStream);

```

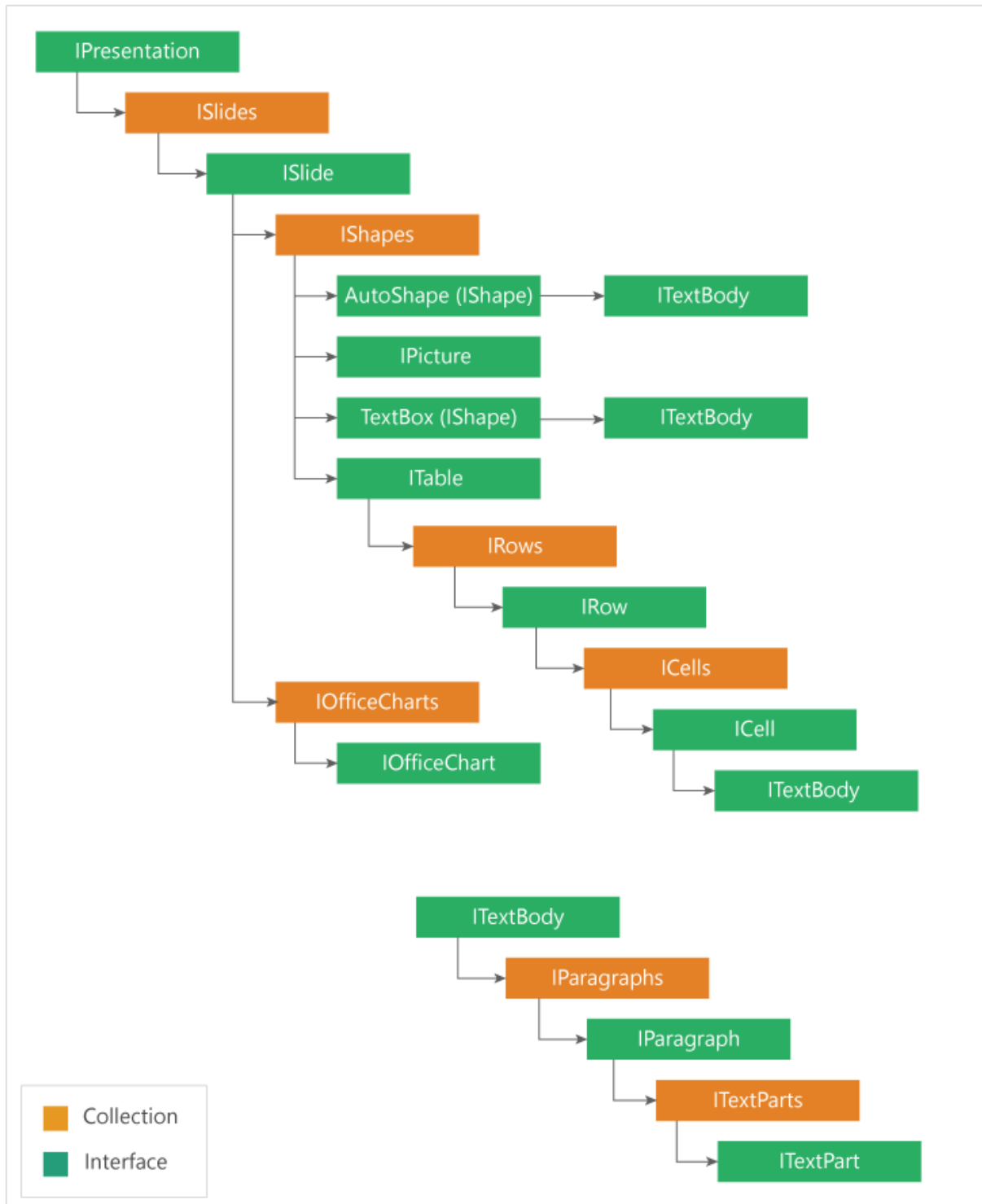
Note: * Creating an instance of **ChartToImageConverter** class is mandatory to convert the charts in the PowerPoint presentation to PDF/Image format. Otherwise, the charts are not exported to the converted PDF/Image.

* **ChartToImageConverter** is supported from .NET Framework 4.0 onwards

PresentationToPdfConverterSettings can be used to customize the conversion of Presentation to PDF document. **ChartToImageConverter** class can be further used to improve the quality of converted charts in the PDF/Image. For more information about this, see [Conversion](#).

Document Object Model representation

In order to create and modify a PowerPoint Presentation, you need to know how the elements are organized in Essential Presentation's document object model (DOM). The following figure illustrates this DOM.



Load and save the Presentation

Opening an existing Presentation from file system

You can open an existing PowerPoint Presentation by using the file name and its physical path.

C#

```
//Opens an existing Presentation from file system  
IPresentation pptxDoc = Presentation.Open(fileName);
```

VB.NET

```
'Opens an existing Presentation from file system  
Dim pptxDoc As IPresentation = Presentation.Open(fileName)
```

UWP

```
//Instantiates the File Picker  
FileOpenPicker openPicker = new FileOpenPicker();  
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;  
openPicker.FileTypeFilter.Add(".pptx");  
//Creates a storage file from FileOpenPicker  
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();  
//Loads or open an PowerPoint Presentation  
IPresentation pptxDoc = await Presentation.OpenAsync(inputStorageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation  
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);  
//Loads or open an PowerPoint Presentation  
IPresentation pptxDoc = Presentation.Open(inputStream)
```

XAMARIN

```
/"App" is the class of Portable project  
Assembly assembly = typeof(App).GetTypeInfo().Assembly;  
Stream inputStream = assembly.GetManifestResourceStream("Sample.pptx");  
//Loads or open an existing PowerPoint Presentation  
IPresentation pptxDoc = Presentation.Open(inputStream);
```

Opening an existing Presentation from stream

You can open an existing PowerPoint Presentation from stream by using the overloads of Open method.

C#

```
//Opens an existing Presentation from stream  
IPresentation pptxDoc = Presentation.Open(presentationStream);
```

VB.NET

```
'Opens an existing Presentation from stream  
Dim pptxDoc As IPresentation = Presentation.Open(presentationStream)
```

UWP

```
//Create new Presentation without slides.  
Assembly assembly = typeof(App).GetTypeInfo().Assembly;  
Stream inputStream = assembly.GetManifestResourceStream(inputFilePath);  
//Loads or open an PowerPoint Presentation  
IPresentation pptxDoc = Presentation.Open(inputStream);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation  
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
```

XAMARIN

```
//Create new Presentation without slides.  
Assembly assembly = typeof(App).GetTypeInfo().Assembly;  
Stream inputStream = assembly.GetManifestResourceStream(inputFilePath);  
//Loads or open an PowerPoint Presentation  
IPresentation pptxDoc = Presentation.Open(inputStream);
```

Opening an encrypted Presentation

You can open an encrypted PowerPoint presentation from either file path or stream by using the following overloads of Open method as follows.

C#

```
//Opens an existing encrypted Presentation from stream  
IPresentation pptxDoc = Presentation.Open(presentationStream, password);
```

VB.NET

```
'Opens an existing encrypted Presentation from stream  
Dim pptxDoc As IPresentation = Presentation.Open(presentationStream,  
password)
```

UWP

```
//Opens an existing encrypted Presentation from stream  
IPresentation pptxDoc = Presentation.OpenAsync(presentationStream,  
password);
```

ASP.NET CORE

```
//Essential Presentation Library does not provides support to Encryption and  
Decryption in ASP.NET Core platforms.
```

XAMARIN

```
//Essential Presentation Library does not provides support to Encryption and  
Decryption in ASP.NET Core platforms.
```

C#

```
//Opens an existing encrypted Presentation from file system  
IPresentation pptxDoc = Presentation.Open(fileName, password);
```

VB.NET

```
'Opens an existing encrypted Presentation from file system  
Dim pptxDoc As IPresentation = Presentation.Open(fileName, password)
```

UWP

```
//Opens an existing encrypted Presentation from file system  
IPresentation pptxDoc = Presentation.OpenAsync(fileName, password);
```

ASP.NET CORE

```
//Essential Presentation Library does not provides support to Encryption and  
Decryption in ASP.NET Core platforms.
```

XAMARIN

```
//Essential Presentation Library does not provides support to Encryption and  
Decryption in ASP.NET Core platforms.
```

Saving a PowerPoint Presentation to file system

You can save the created or manipulated PowerPoint Presentation to file system by using Save() method of **IPresentation** interface. Default format type is *.PPTX.

C#

```
//Opens an existing PowerPoint Presentation  
IPresentation pptxDoc = Presentation.Open(fileName);  
//To-Do some manipulation  
//To-Do some manipulation  
//Saves the Presentation in file system  
pptxDoc.Save("Output.pptx");
```

VB.NET

```
'Opens an existing PowerPoint Presentation  
Dim pptxDoc As IPresentation = Presentation.Open(fileName)  
'To-Do some manipulation  
'To-Do some manipulation  
'Saves the Presentation in file system  
pptxDoc.Save("Output.pptx")
```

UWP

```
//Instantiates the File Picker  
FileOpenPicker openPicker = new FileOpenPicker();  
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;  
openPicker.FileTypeFilter.Add(".pptx");
```

```

//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = await Presentation.OpenAsync(inputStorageFile);
//To-Do some manipulation
//To-Do some manipulation
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(fileName, FileMode.Open);
//To-Do some manipulation
//To-Do some manipulation
FileStream outputStream = new FileStream("output.pptx", FileMode.Create);
pptxDoc.SaveAs(outputStream);

```

XAMARIN

```

//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(inputFilePath);
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//To-Do some manipulation
//To-Do some manipulation
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);

```


Saving a PowerPoint Presentation to stream

You can save the created or manipulated PowerPoint Presentation to stream by using overloads of Save method.

C#

```
//Opens an existing PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(fileName);
//To-Do some manipulation
//To-Do some manipulation
//Creates an instance of memory stream
MemoryStream stream = new MemoryStream();
//Saves the Presentation to stream
pptxDoc.Save(stream);
```

VB.NET

```
'Opens an existing PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open(fileName)
'To-Do some manipulation
'To-Do some manipulation
'Creates an instance of memory stream
Dim stream As New MemoryStream()
'Saves the Presentation to stream
pptxDoc.Save(stream)
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = await Presentation.OpenAsync(inputStorageFile);
//To-Do some manipulation
//To-Do some manipulation
//Saves changes to the specified storage file
MemoryStream outputStream = new MemoryStream();
await pptxDoc.SaveAsync(outputStream);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
//To-Do some manipulation
//To-Do some manipulation
FileStream outputStream = new FileStream(outputFileName, FileMode.Create);
pptxDoc.SaveAs(outputStream);
```

XAMARIN

```
/"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
```

```
Stream inputStream = assembly.GetManifestResourceStream(inputFilePath);
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//To-Do some manipulation
//To-Do some manipulation
MemoryStream outputStream = new MemoryStream();
pptxDoc.Save(outputStream);
```

Sending to a client browser

You can save and send the Presentation to a client browser from a website or web application by invoking the overload of Save method. This method explicitly make use of an instance of `HttpResponse` as its parameter in order to stream the presentation to client browser. So, this overload is suitable for web application that refer to [System.Web](#) assembly.

C#

```
//Opens an existing PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(fileName);
//To-Do some manipulation
//To-Do some manipulation
//Saves the Presentation to the client browser
pptxDoc.Save("Output.pptx", FormatType.Pptx, Response);
```

VB.NET

```
'Opens an existing PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open(fileName)
'To-Do some manipulation
'To-Do some manipulation
'Saves the Presentation to the client browser
pptxDoc.Save("Output.pptx", FormatType.Pptx, Response)
```

UWP

```
//Saving and sending the workbook to a client browser from a web site is
suitable for web applications alone.
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
//To-Do some manipulation
//To-Do some manipulation
//Initialize content type
string ContentType = null;
//Save the PowerPoint Presentation to stream
MemoryStream outputStream = new MemoryStream();
pptxDoc.Save(outputStream);
outputStream.Position = 0;
//Return the file with content type
return File(outputStream, ContentType, outputFileName);
```

XAMARIN

```
//Saving and sending the workbook to a client browser from a web site is  
suitable for web applications alone.
```

Closing a PowerPoint Presentation

When you are done with the Presentation instance, you should close the instance of **IPresentation** in order to release the memory consumed by Essential Presentation library. The following code example illustrates how to close an IPresentation instance.

C#

```
//Opens an existing Presentation from file system  
IPresentation pptxDoc = Presentation.Open(fileName);  
//To-Do some manipulation  
//To-Do some manipulation  
//Creates an instance of memory stream  
MemoryStream stream = new MemoryStream();  
//Saves the Presentation to stream  
pptxDoc.Save(stream);  
//Closes the Presentation instance and free the memory consumed.  
pptxDoc.Close();
```

VB.NET

```
'Opens an existing Presentation from file system  
Dim pptxDoc As IPresentation = Presentation.Open(fileName)  
'To-Do some manipulation  
'To-Do some manipulation  
'Creates an instance of memory stream  
Dim stream As New MemoryStream()  
'Saves the Presentation to stream  
pptxDoc.Save(stream)  
'Closes the Presentation instance and free the memory consumed.  
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker  
FileOpenPicker openPicker = new FileOpenPicker();  
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;  
openPicker.FileTypeFilter.Add(".pptx");  
//Creates a storage file from FileOpenPicker  
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();  
//Loads or open an PowerPoint Presentation  
IPresentation pptxDoc = await Presentation.OpenAsync(inputStorageFile);  
//MemoryStream outputStream = new MemoryStream();  
//await pptxDoc.SaveAsync(outputStream);  
//Initializes FileSavePicker  
FileSavePicker savePicker = new FileSavePicker();  
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;  
savePicker.SuggestedFileName = "Sample";  
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {  
".pptx" });  
//Creates a storage file from FileSavePicker  
StorageFile storageFile = await savePicker.PickSaveFileAsync();  
//Saves changes to the specified storage file
```

```
await pptxDoc.SaveAsync(storageFile);
//Close the instance of PowerPoint Presentation
pptxDoc.Close();
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//To-Do some manipulation
//To-Do some manipulation
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
//Close the instance of PowerPoint Presentation
pptxDoc.Close();
```

XAMARIN

```
///"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(inputFilePath);
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//To-Do some manipulation
//To-Do some manipulation
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Working with PowerPoint presentation

Cloning a PowerPoint presentation

Cloning a PowerPoint presentation creates a new copy of the PowerPoint presentation and the changes made in the cloned copy of the presentation do not affect the source PowerPoint presentation.

C#

```
//Opens a PowerPoint presentation
```

```

IPresentation sourcePresentation = Presentation.Open(fileName);
//Clones the Presentation
IPresentation clonedPresentation = sourcePresentation.Clone();
//Gets the first slide from the cloned PowerPoint presentation
ISlide firstSlide = clonedPresentation.Slides[0];
//Adds a textbox in a slide by specifying its position and size
IShape textShape = firstSlide.AddTextBox(100, 75, 756, 200);
//Adds a paragraph in the body of the textShape
IParagraph paragraph = textShape.TextBody.AddParagraph();
//Adds a textPart in the paragraph
ITextPart textPart = paragraph.AddTextPart("Essential Presentation");
//Saves the modified cloned PowerPoint presentation
clonedPresentation.Save("ClonedPresentation.pptx");

```

VB.NET

```

'Opens a PowerPoint presentation
Dim sourcePresentation_1 As IPresentation = Presentation.Open(fileName)
'Clones the Presentation
Dim clonedPresentation_1 As IPresentation = sourcePresentation_1.Clone()
'Gets the first slide from the cloned PowerPoint presentation
Dim firstSlide As ISlide = clonedPresentation_1.Slides(0)
'Adds a textbox in a slide by specifying its position and size
Dim textShape As IShape = firstSlide.AddTextBox(100, 75, 756, 200)
'Adds a paragraph in the body of the textShape
Dim paragraph As IParagraph = textShape.TextBody.AddParagraph()
'Adds a textPart in the paragraph
Dim textPart As ITextPart = paragraph.AddTextPart("Essential Presentation")
'Saves the modified cloned PowerPoint presentation
clonedPresentation_1.Save("ClonedPresentation.pptx")

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation sourcePresentation = await
Presentation.OpenAsync(inputStorageFile);
//Clones the Presentation
IPresentation clonedPresentation = sourcePresentation.Clone();
//Gets the first slide from the cloned PowerPoint presentation
ISlide firstSlide = clonedPresentation.Slides[0];
//Adds a textbox in a slide by specifying its position and size
IShape textShape = firstSlide.AddTextBox(100, 75, 756, 200);
//Adds a paragraph in the body of the textShape
IParagraph paragraph = textShape.TextBody.AddParagraph();
//Adds a textPart in the paragraph
ITextPart textPart = paragraph.AddTextPart("Essential Presentation");
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ClonedPresentation";

```

```

savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await clonedPresentation.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Clones the Presentation
IPresentation clonedPresentation = pptxDoc.Clone();
String ContentType=null;
//Gets the first slide from the cloned PowerPoint presentation
ISlide firstSlide = clonedPresentation.Slides[0];
//Adds a textbox in a slide by specifying its position and size
IShape textShape = firstSlide.AddTextBox(100, 75, 756, 200);
//Adds a paragraph in the body of the textShape
IParagraph paragraph = textShape.TextBody.AddParagraph();
//Adds a textPart in the paragraph
ITextPart textPart = paragraph.AddTextPart("Essential Presentation");
//Save the PowerPoint Presentation to stream
FileStream outputStream = new FileStream(outputFileName, FileMode.Create);
clonedPresentation.SaveAs(outputStream);

```

XAMARIN

```

//Loads the PowerPoint Presentation as stream.
Assembly assembly =
typeof(GettingStartedPresentation).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(inputFilePath);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Clones the Presentation
IPresentation clonedPresentation = pptxDoc.Clone();
//Gets the first slide from the cloned PowerPoint presentation
ISlide firstSlide = clonedPresentation.Slides[0];
//Adds a textbox in a slide by specifying its position and size
IShape textShape = firstSlide.AddTextBox(100, 75, 756, 200);
//Adds a paragraph in the body of the textShape
IParagraph paragraph = textShape.TextBody.AddParagraph();
//Adds a textPart in the paragraph
ITextPart textPart = paragraph.AddTextPart("Essential Presentation");
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
clonedPresentation.Save(stream);
//Close the presentation
clonedPresentation.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Printing a PowerPoint presentation

You can print the Presentation document by converting the PowerPoint presentation slides to images. For more information about converting the PowerPoint presentation slides to images, see [Conversion](#). You can use the System.Drawing.Printing.[PrintDocument](#) class to print the converted images by the default printer or to any of the available printer with customized settings.

The following code example demonstrates how to convert the slides of a PowerPoint presentation to images.

C#

```
//Opens a PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Converts the slides to images
Image[] images =
pptxDoc.RenderAsImages(Syncfusion.Drawing.ImageType.Bitmap);
//Closes the PowerPoint presentation
pptxDoc.Close();
```

VB.NET

```
'Opens a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Converts the slides to images
Dim images As Image() = pptxDoc.RenderAsImages(Syncfusion.Drawing.
ImageType.Bitmap)
'Closes the PowerPoint presentation
pptxDoc.Close()
```

The following code example demonstrates how to print the converted images.

C#

```
//Initializes the start and page for printing
int startPageIndex = 1;
int endPageIndex = images.Length;
//Creates new PrintDialog instance.
System.Windows.Forms.PrintDialog printDialog = new
System.Windows.Forms.PrintDialog();
//Sets new PrintDocument instance to print dialog.
printDialog.Document = new PrintDocument();
//Enables the print current page option.
printDialog.AllowCurrentPage = true;
//Enables the print selected pages option.
printDialog.AllowSomePages = true;
//Sets the start and end page index
printDialog.PrinterSettings.FromPage = 1;
```

```
printDialog.PrinterSettings.ToPage = images.Length;
//Opens the print dialog box.
if (printDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
{
    //Checks whether the selected page range is valid or not
    if (printDialog.PrinterSettings.FromPage > 0 &&
        printDialog.PrinterSettings.ToPage <= images.Length)
    {
        //Updates the start page of the document to print.
        startPageIndex = printDialog.PrinterSettings.FromPage - 1;
        //Updates the end page of the document to print.
        endPageIndex = printDialog.PrinterSettings.ToPage;
        //Hooks the PrintPage event to handle be drawing pages for printing.
        printDialog.Document.PrintPage += new
        PrintPageEventHandler(PrintPageMethod);
        //Prints the document.
        printDialog.Document.Print();
    }
}

private void PrintPageMethod (object sender, PrintPageEventArgs e)
{
    //Gets the print start page width.
    int currentPageWidth = images[startPageIndex].Width;
    //Gets the print start page height.
    int currentPageHeight = images[startPageIndex].Height;
    //Gets the visible bounds width for print.
    int visibleClipBoundsWidth = (int)e.Graphics.VisibleClipBounds.Width;
    //Gets the visible bounds height for print.
    int visibleClipBoundsHeight = (int)e.Graphics.VisibleClipBounds.Height;
    //Checks whether the page layout is landscape or portrait.
    if (currentPageWidth > currentPageHeight)
    {
        //Translates the position.
        e.Graphics.TranslateTransform(0, visibleClipBoundsHeight);
        //Rotates the object at 270 degrees
        e.Graphics.RotateTransform(270.0f);
        //Draws the current page image.
        e.Graphics.DrawImage(images[startPageIndex], new System.Drawing.Rectangle(0,
        0, currentPageWidth, currentPageHeight));
    }
    else
    {
        //Draws the current page image.
        e.Graphics.DrawImage(images[startPageIndex], new System.Drawing.Rectangle(0,
        0, visibleClipBoundsWidth, visibleClipBoundsHeight));
    }
    //Disposes the current page image after drawing.
    images[startPageIndex].Dispose();
    //Increments the start page index.
    startPageIndex++;
    //Updates whether the document contains more pages to print or not.
    if (startPageIndex < endPageIndex)
    {
        e.HasMorePages = true;
    }
    else
    {
        startPageIndex = 0;
    }
}
```


VB.NET

```

Dim endPageIndex As Integer = images.Length
'Creates new PrintDialog instance.
Dim printDialog As New System.Windows.Forms. PrintDialog()
'Sets new PrintDocument instance to print dialog.
printDialog.Document = New PrintDocument()
'Enables the print current page option.
printDialog.AllowCurrentPage = True
'Enables the print selected pages option.
printDialog.AllowSomePages = True
'Sets the start and end page index
printDialog.PrinterSettings.FromPage = 1
printDialog.PrinterSettings.ToPage = images.Length
'Opens the print dialog box.
If printDialog.ShowDialog() = System.Windows.Forms.DialogResult.OK Then
'Checks whether the selected page range is valid or not.
If printDialog.PrinterSettings.FromPage > 0 AndAlso
printDialog.PrinterSettings.ToPage <= images.Length Then
'Updates the start page of the document to print.
startPageIndex = printDialog.PrinterSettings.FromPage - 1
'Updates the end page of the document to print.
endPageIndex = printDialog.PrinterSettings.ToPage
'Hooks the PrintPage event to handle the drawing pages for printing.
printDialog.Document.PrintPage += New PrintPageEventHandler(PrintPageMethod)
'Prints the document.
printDialog.Document.Print()
End If
End If
Private Sub PrintPageMethod(sender As Object, e As PrintPageEventArgs)
'Gets the print start page width.
Dim currentPageWidth As Integer = images(startPageIndex).Width
'Gets the print start page height.
Dim currentPageHeight As Integer = images(startPageIndex).Height
'Gets the visible bounds width for print.
Dim visibleClipBoundsWidth As Integer =
CInt(e.Graphics.VisibleClipBounds.Width)
'Gets the visible bounds height for print.
Dim visibleClipBoundsHeight As Integer =
CInt(e.Graphics.VisibleClipBounds.Height)
'Checks whether the page layout is landscape or portrait.
If currentPageWidth > currentPageHeight Then
'Translates the position.
e.Graphics.TranslateTransform(0, visibleClipBoundsHeight)
'Rotates the object at 270 degrees.
e.Graphics.RotateTransform(270.0F)
'Draws the current page image.
e.Graphics.DrawImage(images(startPageIndex), New System.Drawing.Rectangle
(0, 0, currentPageWidth, currentPageHeight))
Else
'Draws the current page image.
e.Graphics.DrawImage(images(startPageIndex), New System.Drawing.Rectangle
(0, 0, visibleClipBoundsWidth, visibleClipBoundsHeight))
End If
'Disposes the current page image after drawing.

```

```

images(startPageIndex).Dispose()
'Increments the start page index.
startPageIndex += 1
'Updates whether the document contains more pages to print or not.
If startPageIndex < endPageIndex Then
e.HasMorePages = True
Else
startPageIndex = 0
End If
End Sub

```

Working with PowerPoint presentation properties

Document properties, also known as meta data, are details about a file that describe or identify it. Document properties are classified into two categories.

- **Built-in Document Properties** - that include details such as title, author name, subject, and keywords that identify the document's topic or contents.
- **Custom Document properties** - define the user-defined document properties.

Built-in Document Properties

You can access and modify the built in document properties of a PowerPoint presentation with Essential Presentation library. The Built-in document properties of a PowerPoint presentation is represented by **IBuiltInDocumentProperties** type.

Accessing and Modifying Built-in Document Properties

The following code example demonstrates how to access the existing built in document property.

C#

```

//Opens a PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Accesses the built-in document properties
Console.WriteLine("Title - {0}", pptxDoc.BuiltInDocumentProperties.Title);
Console.WriteLine("Author - {0}", pptxDoc.BuiltInDocumentProperties.Author);
//Closes the PowerPoint presentation
pptxDoc.Close();

```

VB.NET

```

'Opens a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Accesses the built-in document properties
Console.WriteLine("Title - {0}",
presentationDocument.BuiltInDocumentProperties.Title)
Console.WriteLine("Author - {0}",
presentationDocument.BuiltInDocumentProperties.Author)
'Closes the PowerPoint presentation
pptxDoc.Close()

```

The following code example demonstrates how to modify the existing built in document property

C#

```
//Opens a PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Modifies the Built-in document properties
pptxDoc.BuiltInDocumentProperties.Category = "Sales reports";
pptxDoc.BuiltInDocumentProperties.Company = "Northwind traders";
//Saves the modified PowerPoint presentation
pptxDoc.Save("Output.pptx");
//Closes the modified PowerPoint presentation
pptxDoc.Close();
```

VB.NET

```
'Opens a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Modifies the Built-in document properties
pptxDoc.BuiltInDocumentProperties.Category = "Sales reports"
pptxDoc.BuiltInDocumentProperties.Company = "Northwind traders"
'Saves the modified PowerPoint presentation
pptxDoc.Save("Output.pptx")
'Closes the modified PowerPoint presentation
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = await Presentation.OpenAsync(inputStorageFile);
//Modifies the Built-in document properties
pptxDoc.BuiltInDocumentProperties.Category = "Sales reports";
pptxDoc.BuiltInDocumentProperties.Company = "Northwind traders";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Modifies the Built-in document properties
pptxDoc.BuiltInDocumentProperties.Category = "Sales reports";
pptxDoc.BuiltInDocumentProperties.Company = "Northwind traders";
//Save the PowerPoint Presentation as stream
```

```

FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
//Close the instance of PowerPoint Presentation
pptxDoc.Close();

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
// Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
// Modifies the Built-in document properties
pptxDoc.BuiltInDocumentProperties.Category = "Sales reports";
pptxDoc.BuiltInDocumentProperties.Company = "Northwind traders";
// Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
// Save Presentation in stream format.
pptxDoc.Save(stream);
// Close the presentation
pptxDoc.Close();
stream.Position = 0;
// The operation in Save under Xamarin varies between Windows Phone, Android
// and iOS platforms. Please refer presentation/xamarin section for respective
// code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);

```

Custom Document properties

You can create and modify the custom document properties of a PowerPoint presentation with Essential Presentation library. The collection of custom document properties in a PowerPoint presentation is represented by **ICustomDocumentProperties** object.

Adding Custom Document properties

The following code example demonstrates how to add new custom document property.

C#

```

// Creates a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
// Adds custom document properties
ICustomDocumentProperties documentProperty =
    pptxDoc.CustomDocumentProperties;
documentProperty.Add("PropertyA");
documentProperty["PropertyA"].Text = "@!123";
documentProperty.Add("PropertyB");
documentProperty["PropertyB"].Text = "B";
// Saves the PowerPoint presentation

```

```
pptxDoc.Save("Output.pptx");
//Closes the PowerPoint presentation
pptxDoc.Close();
```

VB.NET

```
'Creates a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds custom document properties
Dim documentProperty As ICustomDocumentProperties =
pptxDoc.CustomDocumentProperties
documentProperty.Add("PropertyA")
documentProperty("PropertyA").Text = "@!123"
documentProperty.Add("PropertyB")
documentProperty("PropertyB").Text = "B"
'Saves the PowerPoint presentation
pptxDoc.Save("Output.pptx")
'Closes the PowerPoint presentation
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = await Presentation.OpenAsync(inputStorageFile);
//Adds custom document properties
ICustomDocumentProperties documentProperty =
pptxDoc.CustomDocumentProperties;
documentProperty.Add("PropertyA");
documentProperty["PropertyA"].Text = "@!123";
documentProperty.Add("PropertyB");
documentProperty["PropertyB"].Text = "B";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Adds custom document properties
```

```

ICustomDocumentProperties documentProperty =
pptxDoc.CustomDocumentProperties;
documentProperty.Add("PropertyA");
documentProperty["PropertyA"].Text = "@!123";
documentProperty.Add("PropertyB");
documentProperty["PropertyB"].Text = "B";
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the PowerPoint presentation
pptxDoc.Close();

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
// Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
// Adds custom document properties
ICustomDocumentProperties documentProperty =
pptxDoc.CustomDocumentProperties;
documentProperty.Add("PropertyA");
documentProperty["PropertyA"].Text = "@!123";
documentProperty.Add("PropertyB");
documentProperty["PropertyB"].Text = "B";
// Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
// Save Presentation in stream format.
pptxDoc.Save(stream);
// Close the presentation
pptxDoc.Close();
stream.Position = 0;
// The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Accessing and Modifying Custom Document Properties

The following code example demonstrates how to access and modify an existing custom document property:

C#

```

// Opens a PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
// Accesses an existing custom document property

```

```

IDocumentProperty property = pptxDoc.CustomDocumentProperties["PropertyA"];
//Modifies the value of DocumentProperty
property.Value = "Hello world";
//Saves the PowerPoint presentation
pptxDoc.Save("Output.pptx");
//Closes the PowerPoint presentation
pptxDoc.Close();

```

VB.NET

```

'Opens a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Accesses an existing custom document property
Dim [property] As IDocumentProperty =
pptxDoc.CustomDocumentProperties("PropertyA")
'Modifies the value of DocumentProperty
[property].Value = "Hello world"
'Saves the PowerPoint presentation
pptxDoc.Save("Output.pptx")
'Closes the PowerPoint presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = await Presentation.OpenAsync(inputStorageFile);
//Accesses an existing custom document property
IDocumentProperty property = pptxDoc.CustomDocumentProperties["PropertyA"];
//Modifies the value of DocumentProperty
property.Value = "Hello world";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Accesses an existing custom document property
IDocumentProperty property = pptxDoc.CustomDocumentProperties["PropertyA"];
//Modifies the value of DocumentProperty

```

```
property.Value = "Hello world";
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
```

XAMARIN

```
/"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Accesses an existing custom document property
IDocumentProperty property = pptxDoc.CustomDocumentProperties["PropertyA"];
//Modifies the value of DocumentProperty
property.Value = "Hello world";
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Marking a PowerPoint presentation as final

PowerPoint presentation can be made read-only to prevent the readers from making inadvertent changes to it. However, making presentation as final is not a security feature. Anyone can disable the final status and edit the presentation.

Below code snippet demonstrates how to create a final non – editable presentation,

C#

```
//Create an instance for PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Mark the presentation as final
pptxDoc.Final = true;
//Save the presentation
pptxDoc.Save("MarkAsFinal.pptx");
//Close the presentation
pptxDoc.Close();
```


VB.NET

```
'Create an instance for PowerPoint presentation  
Dim pptxDoc As IPresentation = Presentation.Create()  
'Add slide to the presentation  
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)  
'Mark the presentation as final  
pptxDoc.Final = True  
'Save the presentation  
pptxDoc.Save("MarkAsFinal.pptx")  
'Close the presentation  
pptxDoc.Close()
```

UWP

```
//Loads or open an PowerPoint Presentation  
IPresentation pptxDoc = Presentation.Create();  
//Add slide to the presentation  
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);  
//Mark the presentation as final  
pptxDoc.Final = true;  
//Initializes FileSavePicker  
FileSavePicker savePicker = new FileSavePicker();  
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;  
savePicker.SuggestedFileName = "MarkAsFinal";  
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {  
    ".pptx" });  
//Creates a storage file from FileSavePicker  
StorageFile storageFile = await savePicker.PickSaveFileAsync();  
//Saves changes to the specified storage file  
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Create an instance for PowerPoint presentation  
IPresentation pptxDoc = Presentation.Create();  
//Add slide to the presentation  
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);  
//Mark the presentation as final  
pptxDoc.Final = true;  
//Save the PowerPoint Presentation as stream  
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);  
pptxDoc.Save(outputStream);  
//Close the presentation  
pptxDoc.Close();
```

XAMARIN

```
//Create an instance for PowerPoint presentation  
IPresentation pptxDoc = Presentation.Create();  
//Add slide to the presentation  
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);  
//Mark the presentation as final  
pptxDoc.Final = true;
```

```
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("MarkFinal.ppt
x", "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("MarkFinal.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Working with Slides in PowerPoint

Adding slide to the PowerPoint presentation

In PowerPoint presentation, a slide is a container for the elements like shapes, images, charts, text box etc. The slides may inherit the formatting and layout properties from its 'Master' and 'Layout' slides.

The following code example demonstrates how to add a blank slide to the Presentation.

C#

```
//Creates a PowerPoint instance
IPresentation pptxDoc = Presentation.Create();
//Adds a slide to the PowerPoint presentation
ISlide slide = pptxDoc.Slides.Add();
//Saves the Presentation to the file system.
pptxDoc.Save("Sample.pptx");
//Closes the Presentation instance
pptxDoc.Close();
```

VB.NET

```
'Creates a PowerPoint instance
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds a slide to the PowerPoint presentation
Dim slide As ISlide = pptxDoc.Slides.Add()
'Saves the Presentation to the file system.
pptxDoc.Save("Sample.pptx")
'Closes the Presentation instance
pptxDoc.Close()
```

UWP

```
//Creates a PowerPoint instance
IPresentation pptxDoc = Presentation.Create();
//Adds new Blank type of slide.
ISlide slide = pptxDoc.Slides.Add();
```

```
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Creates a PowerPoint instance
IPresentation pptxDoc = Presentation.Create();
//Adds a slide to the PowerPoint presentation
ISlide slide = pptxDoc.Slides.Add();
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Release all resources of the stream
outputStream.Dispose();
//Closes the Presentation instance
pptxDoc.Close();
```

XAMARIN

```
//Creates a PowerPoint instance
IPresentation pptxDoc = Presentation.Create();
//Adds a slide to the PowerPoint presentation
ISlide slide = pptxDoc.Slides.Add();
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);
```

Create a slide with predefined LayoutSlide

The Syncfusion PowerPoint library supports the following predefined slide layout types to create a slide as equivalent to Microsoft PowerPoint:

```
<ul>
<li>Blank</li>
<li>Comparison</li>
<li>Content with caption</li>
<li>Picture with caption</li>
<li>Section header</li>
<li>Title</li>
<li>Title and content</li>
<li>Title and vertical text</li>
<li>Title only</li>
<li>Two content</li>
<li>Vertical title and text</li>
</ul>
```

The following example demonstrates how to access a slide from the predefined blank slide layout type.

C#

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a slide of blank layout type
ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Save the PowerPoint file
pptxDoc.Save("Sample.pptx");
//Close the PowerPoint instance
pptxDoc.Close();
```

VB.NET

```
'Create a PowerPoint file
Dim pptxDoc As IPresentation = Presentation.Create()
'Add a slide of blank layout type
Dim slide1 As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Save the PowerPoint file
pptxDoc.Save("Sample.pptx")
'Close the PowerPoint instance
pptxDoc.Close()
```

UWP

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a slide of blank layout type
ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
```

```
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Create a new instance of PowerPoint Presentation file
IPresentation pptxDoc = Presentation.Create();
//Add a slide of blank layout type
ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
//Release all resources of the stream
outputStream.Dispose();
//Close the PowerPoint presentation
pptxDoc.Close();
```

XAMARIN

```
//Create a new instance of PowerPoint Presentation file
IPresentation pptxDoc = Presentation.Create();
//Add a slide of blank layout type
ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

The following code example demonstrates how to add a slide with all other predefined slide layout types.

C#

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a slide of blank layout type
```

```

ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a slide of comparison type
ISlide slide2 = pptxDoc.Slides.Add(SlideLayoutType.Comparison);
//Add a slide of ContentWithCaption type
ISlide slide3 = pptxDoc.Slides.Add(SlideLayoutType.ContentWithCaption);
//Add a slide of PictureWithCaption type
ISlide slide4 = pptxDoc.Slides.Add(SlideLayoutType.PictureWithCaption);
//Add a slide of SectionHeader type
ISlide slide5 = pptxDoc.Slides.Add(SlideLayoutType.SectionHeader);
//Add a slide of Title type
ISlide slide6 = pptxDoc.Slides.Add(SlideLayoutType.Title);
//Add a slide of TitleAndContent type
ISlide slide7 = pptxDoc.Slides.Add(SlideLayoutType.TitleAndContent);
//Add a slide of TitleAndVerticalText type
ISlide slide8 = pptxDoc.Slides.Add(SlideLayoutType.TitleAndVerticalText);
//Add a slide of TitleOnly type
ISlide slide9 = pptxDoc.Slides.Add(SlideLayoutType.TitleOnly);
//Add a slide of TwoContent type
ISlide slide10 = pptxDoc.Slides.Add(SlideLayoutType.TwoContent);
//Add a slide of VerticalTitleAndText type
ISlide slide11 = pptxDoc.Slides.Add(SlideLayoutType.VerticalTitleAndText);
//Save the PowerPoint file
pptxDoc.Save("Sample.pptx");
//Close the PowerPoint instance
pptxDoc.Close();

```

VB.NET

```

'Create a PowerPoint file
Dim pptxDoc As IPresentation = Presentation.Create()
'Add a slide of blank layout type
Dim slide1 As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a slide of comparison type
Dim ISlide As slide2 = pptxDoc.Slides.Add(SlideLayoutType.Comparison)
'Add a slide of ContentWithCaption type
Dim ISlide As slide3 =
pptxDoc.Slides.Add(SlideLayoutType.ContentWithCaption)
'Add a slide of PictureWithCaption type
Dim ISlide As slide4 =
pptxDoc.Slides.Add(SlideLayoutType.PictureWithCaption)
'Add a slide of SectionHeader type
Dim ISlide As slide5 = pptxDoc.Slides.Add(SlideLayoutType.SectionHeader)
'Add a slide of Title type
ISlide As slide6 = pptxDoc.Slides.Add(SlideLayoutType.Title)
'Add a slide of TitleAndContent type
Dim ISlide As slide7 = pptxDoc.Slides.Add(SlideLayoutType.TitleAndContent)
'Add a slide of TitleAndVerticalText type
Dim ISlide As slide8 =
pptxDoc.Slides.Add(SlideLayoutType.TitleAndVerticalText)
'Add a slide of TitleOnly type
Dim ISlide As slide9 = pptxDoc.Slides.Add(SlideLayoutType.TitleOnly)
'Add a slide of TwoContent type
Dim ISlide As slide10 = pptxDoc.Slides.Add(SlideLayoutType.TwoContent)
'Add a slide of VerticalTitleAndText type
Dim ISlide As slide11 =
pptxDoc.Slides.Add(SlideLayoutType.VerticalTitleAndText)

```

```
'Save the PowerPoint file
pptxDoc.Save("Sample.pptx")
'Close the PowerPoint instance
pptxDoc.Close()
```

UWP

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a slide of blank layout type
ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a slide of comparison type
ISlide slide2 = pptxDoc.Slides.Add(SlideLayoutType.Comparison);
//Add a slide of ContentWithCaption type
ISlide slide3 = pptxDoc.Slides.Add(SlideLayoutType.ContentWithCaption);
//Add a slide of PictureWithCaption type
ISlide slide4 = pptxDoc.Slides.Add(SlideLayoutType.PictureWithCaption);
//Add a slide of SectionHeader type
ISlide slide5 = pptxDoc.Slides.Add(SlideLayoutType.SectionHeader);
//Add a slide of Title type
ISlide slide6 = pptxDoc.Slides.Add(SlideLayoutType.Title);
//Add a slide of TitleAndContent type
ISlide slide7 = pptxDoc.Slides.Add(SlideLayoutType.TitleAndContent);
//Add a slide of TitleAndVerticalText type
ISlide slide8 = pptxDoc.Slides.Add(SlideLayoutType.TitleAndVerticalText);
//Add a slide of TitleOnly type
ISlide slide9 = pptxDoc.Slides.Add(SlideLayoutType.TitleOnly);
//Add a slide of TwoContent type
ISlide slide10 = pptxDoc.Slides.Add(SlideLayoutType.TwoContent);
//Add a slide of VerticalTitleAndText type
ISlide slide11 = pptxDoc.Slides.Add(SlideLayoutType.VerticalTitleAndText);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Create a new instance of PowerPoint Presentation file
IPresentation pptxDoc = Presentation.Create();
//Add a slide of Blank type
ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a slide of comparison type
ISlide slide2 = pptxDoc.Slides.Add(SlideLayoutType.Comparison);
//Add a slide of ContentWithCaption type
ISlide slide3 = pptxDoc.Slides.Add(SlideLayoutType.ContentWithCaption);
//Add a slide of PictureWithCaption type
ISlide slide4 = pptxDoc.Slides.Add(SlideLayoutType.PictureWithCaption);
//Add a slide of SectionHeader type
ISlide slide5 = pptxDoc.Slides.Add(SlideLayoutType.SectionHeader);
```

```
//Add a slide of Title type
ISlide slide6 = pptxDoc.Slides.Add(SlideLayoutType.Title);
//Add a slide of TitleAndContent type
ISlide slide7 = pptxDoc.Slides.Add(SlideLayoutType.TitleAndContent);
//Add a slide of TitleAndVerticalText type
ISlide slide8 = pptxDoc.Slides.Add(SlideLayoutType.TitleAndVerticalText);
//Add a slide of TitleOnly type
ISlide slide9 = pptxDoc.Slides.Add(SlideLayoutType.TitleOnly);
//Add a slide of TwoContent type
ISlide slide10 = pptxDoc.Slides.Add(SlideLayoutType.TwoContent);
//Add a slide of VerticalTitleAndText type
ISlide slide11 = pptxDoc.Slides.Add(SlideLayoutType.VerticalTitleAndText);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
//Release all resources of the stream
outputStream.Dispose();
//Close the PowerPoint presentation
pptxDoc.Close();
```

XAMARIN

```
//Create a new instance of PowerPoint Presentation file
IPresentation pptxDoc = Presentation.Create();
//Add a slide of Blank type
ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a slide of comparison type
ISlide slide2 = pptxDoc.Slides.Add(SlideLayoutType.Comparison);
//Add a slide of ContentWithCaption type
ISlide slide3 = pptxDoc.Slides.Add(SlideLayoutType.ContentWithCaption);
//Add a slide of PictureWithCaption type
ISlide slide4 = pptxDoc.Slides.Add(SlideLayoutType.PictureWithCaption);
//Add a slide of SectionHeader type
ISlide slide5 = pptxDoc.Slides.Add(SlideLayoutType.SectionHeader);
//Add a slide of Title type
ISlide slide6 = pptxDoc.Slides.Add(SlideLayoutType.Title);
//Add a slide of TitleAndContent type
ISlide slide7 = pptxDoc.Slides.Add(SlideLayoutType.TitleAndContent);
//Add a slide of TitleAndVerticalText type
ISlide slide8 = pptxDoc.Slides.Add(SlideLayoutType.TitleAndVerticalText);
//Add a slide of TitleOnly type
ISlide slide9 = pptxDoc.Slides.Add(SlideLayoutType.TitleOnly);
//Add a slide of TwoContent type
ISlide slide10 = pptxDoc.Slides.Add(SlideLayoutType.TwoContent);
//Add a slide of VerticalTitleAndText type
ISlide slide11 = pptxDoc.Slides.Add(SlideLayoutType.VerticalTitleAndText);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
```



```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Adding Custom layout slide

The slide layout are template design for the PowerPoint slides. Slide layout can contains formatting, positioning, and placeholders for a slide. There are 9 predefined layouts and custom slide layouts can also be designed.

The following code example demonstrates how to create and use a customized slide layout in PowerPoint presentation.

C#

```

//Open the template presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Add a new custom layout slide to the master collection with a specific
layout type and name
ILayoutSlide layoutSlide =
pptxDoc.Masters[0].LayoutSlides.Add(SlideLayoutType.Blank, "CustomLayout");
//Set background of the layout slide
layoutSlide.Background.Fill.SolidFill.Color = ColorObject.FromArgb(78, 89,
90);
//Get the stream of an image
Stream pictureStream = File.Open("Image.png", FileMode.Open);
//Add the picture into layout slide
layoutSlide.Shapes.AddPicture(pictureStream, 100, 100, 100, 100);
//Add a slide of new designed custom layout to the presentation
ISlide slide = pptxDoc.Slides.Add(layoutSlide);
//Save the presentation
pptxDoc.Save("Output.pptx");
//Close the presentation
pptxDoc.Close();

```

VB.NET

```

'Open the template presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Add a new custom layout slide to the master collection with a specific
layout type and name
Dim layoutSlide As ILayoutSlide =
pptxDoc.Masters(0).LayoutSlides.Add(SlideLayoutType.Blank, "CustomLayout")
'Set background of the layout slide
layoutSlide.Background.Fill.SolidFill.Color = ColorObject.FromArgb(78, 89,
90)
'Get the stream of an image
Dim pictureStream As Stream = File.Open("Image.png", FileMode.Open)
'Add the picture into layout slide
layoutSlide.Shapes.AddPicture(pictureStream, 100, 100, 100, 100)

```

```

'Add a slide of new designed custom layout to the presentation
Dim slide As ISlide = pptxDoc.Slides.Add(layoutSlide)
'Save the presentation
pptxDoc.Save("Output.pptx")
'Close the presentation
pptxDoc.Close()

```

UWP

```

FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Add a new custom layout slide to the master collection with a specific layout type and name
ILayoutSlide layoutSlide =
pptxDoc.Masters[0].LayoutSlides.Add(SlideLayoutType.Blank, "CustomLayout");
//Set background of the layout slide
layoutSlide.Background.Fill.SolidFill.Color = ColorObject.FromArgb(78, 89, 90);
//Get the stream of an image
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream pictureStream =
assembly.GetManifestResourceStream("UWP.Data.Image.png");
//Add the picture into layout slide
layoutSlide.Shapes.AddPicture(pictureStream, 100, 100, 100, 100);
//Add a slide of new designed custom layout to the presentation
ISlide slide = pptxDoc.Slides.Add(layoutSlide);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Add a new custom layout slide to the master collection with a specific layout type and name
ILayoutSlide layoutSlide =
pptxDoc.Masters[0].LayoutSlides.Add(SlideLayoutType.Blank, "CustomLayout");
//Set background of the layout slide
layoutSlide.Background.Fill.SolidFill.Color = ColorObject.FromArgb(78, 89, 90);
//Get the stream of an image
FileStream pictureStream = new FileStream(inputImagePath, FileMode.Open);

```

```
//Add the picture into layout slide
layoutSlide.Shapes.AddPicture(pictureStream, 100, 100, 100, 100);
//Add a slide of new designed custom layout to the presentation
ISlide slide = pptxDoc.Slides.Add(layoutSlide);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
//Release all resources of the stream
outputStream.Dispose();
//Close the presentation
pptxDoc.Close();
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Add a new custom layout slide to the master collection with a specific
//layout type and name
ILayoutSlide layoutSlide =
pptxDoc.Masters[0].LayoutSlides.Add(SlideLayoutType.Blank, "CustomLayout");
//Set background of the layout slide
layoutSlide.Background.Fill.SolidFill.Color = ColorObject.FromArgb(78, 89,
90);
//Get the stream of an image
Stream pictureStream = assembly.GetManifestResourceStream(picturePath);
//Add the picture into layout slide
layoutSlide.Shapes.AddPicture(pictureStream, 100, 100, 100, 100);
//Add a slide of new designed custom layout to the presentation
ISlide slide = pptxDoc.Slides.Add(layoutSlide);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
//and iOS platforms. Please refer presentation/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

The following code example demonstrates how to add a slide with an existing slide's layout.

C#

```

//Open the template presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Get the layout slide collection of the master
ILayoutSlides layoutSlides = pptxDoc.Masters[0].LayoutSlides;
ILayoutSlide slideLayout = null;
//Get each layout slide from the collection
foreach (ILayoutSlide layout in layoutSlides)
{
    //Check if the layout slide has desired custom layout name
    if (layout.Name == "CustomSlideLayout")
    {
        slideLayout = layout;
        break;
    }
}
//Add slide with the desired layout.
ISlide slide = pptxDoc.Slides.Add(slideLayout);
//Save the presentation
pptxDoc.Save("Output.pptx");
//Close the presentation
pptxDoc.Close();

```

VB.NET

```

'Open the template presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Get the layout slide collection of the master
Dim layoutSlides As ILayoutSlides = pptxDoc.Masters(0).LayoutSlides
Dim slideLayout As ILayoutSlide = Nothing
'Get each layout slide from the collection
For Each layout As ILayoutSlide In layoutSlides
    'Check if the layout slide has desired custom layout name
    If layout.Name = "CustomSlideLayout" Then
        slideLayout = layout
    End If
Next
'Add slide with the desired layout.
Dim slide As ISlide = pptxDoc.Slides.Add(slideLayout)
'Save the presentation
pptxDoc.Save("Output.pptx")
'Close the presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Add a new custom layout slide to the master collection with a specific layout type and name

```

```

ILayoutSlide layoutSlide =
pptxDoc.Masters[0].LayoutSlides.Add(SlideLayoutType.Blank,
"CustomSlideLayout");
//Get the layout slide collection of the master
ILayoutSlides layoutSlides = pptxDoc.Masters[0].LayoutSlides;
ILayoutSlide slideLayout = null;
//Get each layout slide from the collection
foreach (ILayoutSlide layout in layoutSlides)
{
//Check if the layout slide has desired custom layout name
if (layout.Name == "CustomSlideLayout")
{
slideLayout = layout;
break;
}
}
//Add slide with the desired layout.
ISlide slide = pptxDoc.Slides.Add(slideLayout);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get the layout slide collection of the master
ILayoutSlides layoutSlides = pptxDoc.Masters[0].LayoutSlides;
ILayoutSlide slideLayout = null;
//Get each layout slide from the collection
foreach (ILayoutSlide layout in layoutSlides)
{
//Check if the layout slide has desired custom layout name
if (layout.Name == "CustomSlideLayout")
{
slideLayout = layout;
break;
}
}
//Add slide with the desired layout.
ISlide slide = pptxDoc.Slides.Add(slideLayout);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
//Release all resources of the stream
outputStream.Dispose();
//Close the presentation
pptxDoc.Close();

```

XAMARIN

```

///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get the layout slide collection of the master
ILayoutSlides layoutSlides = pptxDoc.Masters[0].LayoutSlides;
ILayoutSlide slideLayout = null;
//Get each layout slide from the collection
foreach (ILayoutSlide layout in layoutSlides)
{
    //Check if the layout slide has desired custom layout name
    if (layout.Name == "CustomSlideLayout")
    {
        slideLayout = layout;
        break;
    }
}
//Add slide with the desired layout.
ISlide slide = pptxDoc.Slides.Add(slideLayout);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);

```

Cloning slide

You can create a deep copy of a slide by cloning the slide. The cloned slide is an independent copy of its source slide. This means the changes made in the cloned slide do not affect the source slide.

C#

```

//Opens an existing Presentation.
IPresentation pptxDoc = Presentation.Open("Presentation.pptx");
//Retrieves the slide instance.
ISlide slide = pptxDoc.Slides[0];
//Creates a cloned copy of slide.
ISlide slideClone = slide.Clone();

```

```
//Adds a new text box to the cloned slide.
IShape textboxShape = slideClone.AddTextBox(0, 0, 250, 250);
//Adds a paragraph with text content to the shape.
textboxShape.TextBody.AddParagraph("Hello Presentation");
//Adds the slide to the Presentation.
pptxDoc.Slides.Add(slideClone);
//Saves the Presentation to the file system.
pptxDoc.Save("Output.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Opens an existing Presentation.
Dim pptxDoc As IPresentation = Presentation.Open("Presentation.pptx")
'Retrieves the slide instance.
Dim slide As ISlide = pptxDoc.Slides(0)
'Creates a cloned copy of slide.
Dim slideClone As ISlide = slide.Clone()
'Adds a new text box to the cloned slide.
Dim textboxShape As IShape = slideClone.AddTextBox(0, 0, 250, 250)
'Adds a paragraph with text content to the shape.
textboxShape.TextBody.AddParagraph("Hello Presentation")
'Adds the slide to the Presentation.
pptxDoc.Slides.Add(slideClone)
'Saves the Presentation to the file system.
pptxDoc.Save("Output.pptx")
'Closes the Presentation
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieves the slide instance.
ISlide slide = pptxDoc.Slides[0];
//Creates a cloned copy of slide.
ISlide slideClone = slide.Clone();
//Adds a new text box to the cloned slide.
IShape textboxShape = slideClone.AddTextBox(0, 0, 250, 250);
//Adds a paragraph with text content to the shape.
textboxShape.TextBody.AddParagraph("Hello Presentation");
//Adds the slide to the Presentation.
pptxDoc.Slides.Add(slideClone);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
```

```
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the slide instance.
ISlide slide = pptxDoc.Slides[0];
//Creates a cloned copy of slide.
ISlide slideClone = slide.Clone();
//Adds a new text box to the cloned slide.
IShape textboxShape = slideClone.AddTextBox(0, 0, 250, 250);
//Adds a paragraph with text content to the shape.
textboxShape.TextBody.AddParagraph("Hello Presentation");
//Adds the slide to the Presentation.
pptxDoc.Slides.Add(slideClone);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
//Release all resources of the stream
outputStream.Dispose();
//Close the presentation
pptxDoc.Close();
```

XAMARIN

```
///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the slide instance.
ISlide slide = pptxDoc.Slides[0];
//Creates a cloned copy of slide.
ISlide slideClone = slide.Clone();
//Adds a new text box to the cloned slide.
IShape textboxShape = slideClone.AddTextBox(0, 0, 250, 250);
//Adds a paragraph with text content to the shape.
textboxShape.TextBody.AddParagraph("Hello Presentation");
//Adds the slide to the Presentation.
pptxDoc.Slides.Add(slideClone);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
```



```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Merging slide

The Essential Presentation provides ability to clone slides from one Presentation to another Presentation. With this ability, you can split a large Presentation into small ones and also merge multiple presentations to one Presentation. You can choose the theme for the cloned slide by using the enum PasteOption.

C#

```

//Opens the source Presentation
IPresentation sourcePresentation =
Presentation.Open("SourcePresentation.pptx");
//Opens the destination Presentation
IPresentation destinationPresentation =
Presentation.Open("DestinationPresentation.pptx");
//Clones the first slide of the source Presentation
ISlide clonedSlide = sourcePresentation.Slides[0].Clone();
//Merges the cloned slide to the destination Presentation with paste option
- Destination Theme
destinationPresentation.Slides.Add(clonedSlide,
PasteOptions.UseDestinationTheme, sourcePresentation);
//Saves the destination Presentation
destinationPresentation.Save("Output.pptx");
//Closes the source presentation
sourcePresentation.Close();
//Closes the destination Presentation
destinationPresentation.Close();

```

VB.NET

```

'Opens the source Presentation
Dim sourcePresentation_1 As IPresentation =
Presentation.Open("SourcePresentation.pptx")
'Opens the destination Presentation
Dim destinationPresentation_1 As IPresentation =
Presentation.Open("DestinationPresentation.pptx")
'Clones the first slide of the source Presentation
Dim clonedSlide As ISlide = sourcePresentation_1.Slides(0).Clone()
'Merges the cloned slide to the destination Presentation with paste option -
Destination Theme
destinationPresentation_1.Slides.Add(clonedSlide,
PasteOptions.UseDestinationTheme, sourcePresentation_1)
'Saves the destination Presentation
destinationPresentation_1.Save("Output.pptx")
'Closes the source presentation
sourcePresentation.Close()

```

```
'Closes the destination Presentation
destinationPresentation.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile sourceStorageFile = await openPicker.PickSingleFileAsync();
//Opens the source Presentation
IPresentation sourcePresentation = await
Presentation.OpenAsync(sourceStorageFile);
//Creates a storage file from FileOpenPicker
StorageFile destinationStorageFile = await openPicker.PickSingleFileAsync();
//Opens the destination Presentation
IPresentation destinationPresentation = await
Presentation.OpenAsync(destinationStorageFile);
//Clones the first slide of the source Presentation
ISlide clonedSlide = sourcePresentation.Slides[0].Clone();
//Merges the cloned slide to the destination Presentation with paste option
- Destination Theme
destinationPresentation.Slides.Add(clonedSlide,
PasteOptions.UseDestinationTheme, sourcePresentation);
//Saves the destination Presentation
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await destinationPresentation.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Opens the source Presentation
IPresentation sourcePresentation =
Presentation.Open(SourcePresentationStream);
//Opens the destination Presentation
IPresentation destinationPresentation =
Presentation.Open(destinationPresentationStream);
//Clones the first slide of the source Presentation
ISlide clonedSlide = sourcePresentation.Slides[0].Clone();
//Merges the cloned slide to the destination Presentation with paste option
- Destination Theme
destinationPresentation.Slides.Add(clonedSlide,
PasteOptions.UseDestinationTheme, sourcePresentation);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
destinationPresentation.Save(outputStream);
//Release all resources of the stream
outputStream.Dispose();
```

```
//Closes the source presentation
sourcePresentation.Close();
//Closes the destination Presentation
destinationPresentation.Close();
```

XAMARIN

```
/"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream(sourcePresentationPath);
Stream outputStream =
assembly.GetManifestResourceStream(destinationPresentationPath);
//Loads or open an PowerPoint Presentation
IPresentation sourcePresentation = Presentation.Open(inputStream);
//Loads or open an PowerPoint Presentation
IPresentation destinationPresentation = Presentation.Open(outputStream);
//Clones the first slide of the source Presentation
ISlide clonedSlide = sourcePresentation.Slides[0].Clone();
//Merges the cloned slide to the destination Presentation with paste option
- Destination Theme
destinationPresentation.Slides.Add(clonedSlide,
PasteOptions.UseDestinationTheme, sourcePresentation);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
destinationPresentation.Save(stream);
//Close the presentation
destinationPresentation.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Removing slide

The Essential Presentation provides the ability to delete a slide by its instance or by its index position in slide collection. The following code demonstrates how to delete a slide from a presentation.

C#

```
//Opens an existing presentation.
IPresentation pptxDoc = Presentation.Open("Presentation1.pptx");
//Retrieves the slide instance.
ISlide slide = pptxDoc.Slides[0];
//Removes the specified slide from the Presentation.
pptxDoc.Slides.Remove(slide);
```

```
// Removes the slide from the specified index.
pptxDoc.Slides.RemoveAt(1);
//Saves the destination Presentation
pptxDoc.Save("Output.pptx");
//Closes the Presentation instance
pptxDoc.Close();
```

VB.NET

```
'Opens an existing Presentation.
Dim pptxDoc As IPresentation = Presentation.Open("Presentation1.pptx")
'Retrieves the slide instance.
Dim slide As ISlide = pptxDoc.Slides(0)
'Removes the specified slide from the Presentation.
pptxDoc.Slides.Remove(slide)
'Removes the slide from the specified index.
pptxDoc.Slides.RemoveAt(1)
'Saves the destination Presentation
pptxDoc.Save("Output.pptx")
'Closes the Presentation instance
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieves the slide instance.
ISlide slide = pptxDoc.Slides[0];
//Removes the specified slide from the Presentation.
pptxDoc.Slides.Remove(slide);
//Removes the slide from the specified index.
pptxDoc.Slides.RemoveAt(1);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the slide instance.
```

```

ISlide slide = pptxDoc.Slides[0];
//Removes the specified slide from the Presentation.
pptxDoc.Slides.Remove(slide);
// Removes the slide from the specified index.
pptxDoc.Slides.RemoveAt(1);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
//Release all resources of the stream
outputStream.Dispose();
//Closes the Presentation instance
pptxDoc.Close();

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the slide instance.
ISlide slide = pptxDoc.Slides[0];
//Removes the specified slide from the Presentation.
pptxDoc.Slides.Remove(slide);
// Removes the slide from the specified index.
pptxDoc.Slides.RemoveAt(1);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Converting to image

You can convert a presentation slide to image with Essential Presentation. The following code example converts the first slide of a PowerPoint presentation into image and saves the image to a file.

C#

```

//Opens a PowerPoint presentation file
IPresentation pptxDoc = Presentation.Open(fileName);

```

```
//Creates an instance of ChartToImageConverter and assigns it to
ChartToImageConverter
pptxDoc.ChartToImageConverter = new ChartToImageConverter();
//Converts the first slide into image
Image image =
pptxDoc.Slides[0].ConvertToImage(Syncfusion.Drawing.ImageType.Metafile);
//Saves the image as file
image.Save("slide1.png");
//Closes the Presentation instance
pptxDoc.Close();
```

VB.NET

```
'Opens a PowerPoint presentation file
Dim pptxDoc As IPresentation = Presentation.Open(fileName)
'Creates an instance of ChartToImageConverter and assigns it to
ChartToImageConverter
pptxDoc.ChartToImageConverter = New ChartToImageConverter()
'Converts the first slide into image
Dim image As Image =
pptxDoc.Slides(0).ConvertToImage(Syncfusion.Drawing.ImageType.Metafile)
'Saves the image as file
image.Save("slide1.png")
'Closes the Presentation instance
pptxDoc.Close()
```

UWP

```
//Load the presentation file using open picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.FileTypeFilter.Add(".pptx");
StorageFile inputFile = await openPicker.PickSingleFileAsync();
IPresentation pptxDoc = await Presentation.OpenAsync(inputFile);
//Pick the folder to save the converted images.
FolderPicker folderPicker = new FolderPicker();
folderPicker.ViewMode = PickerViewMode.Thumbnail;
folderPicker.FileTypeFilter.Add("*");
StorageFolder storageFolder = await folderPicker.PickSingleFolderAsync();
//Create a cancellation token to cancel the image rendering instantly.
CancellationTokenSource cancellationToken = new CancellationTokenSource();
//Convert the slide to image.
ISlide slidel = pptxDoc.Slides[0];
StorageFile imageFile = await storageFolder.CreateFileAsync("Slide1.png",
CreationCollisionOption.ReplaceExisting);
await slidel.SaveAsImageAsync(imageFile, cancellationToken.Token);
//Close the presentation instance
pptxDoc.Close();
```

ASP.NET CORE

```
//PowerPoint Presentation to image conversion is not supported for ASP.NET
Core platforms.
```

XAMARIN

```
//PowerPoint Presentation to image conversion is not supported for Xamarin platforms.
```

For more details on assemblies required for converting a slide to image, see [Conversion](#)

Note: You can print the PowerPoint presentations by using its ability to convert the slides as images. For more details, refer to [Printing a PowerPoint presentation](#)

Changing Slide background

The following code example demonstrates setting the background for a slide.

C#

```
//Opens an existing Presentation.
IPresentation pptxDoc = Presentation.Open("Presentation1.pptx");
//Retrieves the slide instance.
ISlide slide = pptxDoc.Slides[0];
//Retrieves the background instance.
IBackground background = slide.Background;
//Sets the fill type of the background to gradient.
background.Fill.FillType = FillType.Gradient;
//Retrieves the fill of the background to the IGradientFill instance.
IGradientFill gradient = background.Fill.GradientFill;
//Adds the first gradient stop of the gradient fill.
gradient.GradientStops.Add(ColorObject.Green, 20);
//Adds the second gradient stop of the gradient fill.
gradient.GradientStops.Add(ColorObject.Yellow, 50);
//Saves the Presentation to the file system
pptxDoc.Save("Output.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Opens an existing Presentation.
Dim pptxDoc As IPresentation = Presentation.Open("Presentation1.pptx")
'Retrieves the slide instance.
Dim slide As ISlide = pptxDoc.Slides(0)
'Retrieves the background instance.
Dim background As IBackground = slide.Background
'Sets the fill type of the background to gradient.
background.Fill.FillType = FillType.Gradient
'Retrieves the fill of the background to the IGradientFill instance.
Dim gradient As IGradientFill = background.Fill.GradientFill
'Adds the first gradient stop of the gradient fill.
gradient.GradientStops.Add(ColorObject.Green, 20)
'Adds the second gradient stop of the gradient fill.
gradient.GradientStops.Add(ColorObject.Yellow, 50)
'Saves the Presentation to the file system
pptxDoc.Save("Output.pptx")
'Closes the Presentation
pptxDoc.Close()
```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieves the slide instance.
ISlide slide = pptxDoc.Slides[0];
//Retrieves the background instance.
IBackground background = slide.Background;
//Sets the fill type of the background to gradient.
background.Fill.FillType = FillType.Gradient;
//Retrieves the fill of the background to the IGradientFill instance.
IGradientFill gradient = background.Fill.GradientFill;
//Adds the first gradient stop of the gradient fill.
gradient.GradientStops.Add(ColorObject.Green, 20);
//Adds the second gradient stop of the gradient fill.
gradient.GradientStops.Add(ColorObject.Yellow, 50);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the slide instance.
ISlide slide = pptxDoc.Slides[0];
//Retrieves the background instance.
IBackground background = slide.Background;
//Sets the fill type of the background to gradient.
background.Fill.FillType = FillType.Gradient;
//Retrieves the fill of the background to the IGradientFill instance.
IGradientFill gradient = background.Fill.GradientFill;
//Adds the first gradient stop of the gradient fill.
gradient.GradientStops.Add(ColorObject.Green, 20);
//Adds the second gradient stop of the gradient fill.
gradient.GradientStops.Add(ColorObject.Yellow, 50);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
//Release all resources of the stream
outputStream.Dispose();
//Closes the Presentation
pptxDoc.Close();

```


XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
// Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
// Retrieves the slide instance.
ISlide slide = pptxDoc.Slides[0];
// Retrieves the background instance.
IBackground background = slide.Background;
// Sets the fill type of the background to gradient.
background.Fill.FillType = FillType.Gradient;
// Retrieves the fill of the background to the IGradientFill instance.
IGradientFill gradient = background.Fill.GradientFill;
// Adds the first gradient stop of the gradient fill.
gradient.GradientStops.Add(ColorObject.Green, 20);
// Adds the second gradient stop of the gradient fill.
gradient.GradientStops.Add(ColorObject.Yellow, 50);
// Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
// Save Presentation in stream format.
pptxDoc.Save(stream);
// Close the presentation
pptxDoc.Close();
stream.Position = 0;
// The operation in Save under Xamarin varies between Windows Phone, Android
// and iOS platforms. Please refer presentation/xamarin section for respective
// code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);

```

Create and edit Master and Layout slides

To get all the slides in same format, you should perform those changes in the Slide Master or Layout Master. The changes will be applied to all the slides, which inherits the master slide or layout slide.

The [Syncfusion PowerPoint library](#) supports the following:

-
- Access the MasterSlide in PowerPoint file.
- Add LayoutSlide to the MasterSlide.
- Customize the LayoutSlide.
- Create a slide with 9 pre-defined layout slides.
- Customize the layout slides to fit your own scenarios.
-

Access the MasterSlide

In PowerPoint presentation, the **MasterSlide** is the top slide that controls all information about the theme, layout, background, color, fonts, and positioning of all slides. Using this MasterSlide, you can easily adjust the look of an existing theme or make overall changes to all your slides.

The following code example demonstrates how to access the **MasterSlide** in a PowerPoint presentation.

C#

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Access the first master slide in PowerPoint file
IMasterSlide masterSlide = pptxDoc.Masters[0];
//Get the first shape name from the master slide
string shapeName = masterSlide.Shapes[0].ShapeName;
//Save the PowerPoint file
pptxDoc.Save("Sample.pptx");
//Close the Presentation instance
pptxDoc.Close();
```

VB.NET

```
'Create a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Access the first master slide in PowerPoint file.
Dim masterSlide As IMasterSlide = pptxDoc.Masters(0)
'Get the first shape name from the master slide
Dim shapeName As String = masterSlide.Shapes(0).ShapeName
'Save the PowerPoint file.
pptxDoc.Save("AccessMasterSlide.pptx")
'Close the Presentation instance
pptxDoc.Close()
```

Create a custom LayoutSlide

The real-world scenarios always require more predefined templates. The Syncfusion PowerPoint library lets you build your own custom layout designs and use them to create individual slides.

The following code example demonstrates how to create new custom layout slide and access layout slide in Presentation.

C#

```
//Create a PowerPoint instance
IPresentation pptxDoc = Presentation.Create();
//Add a new LayoutSlide to the PowerPoint file
ILayoutSlide layoutSlide =
pptxDoc.Masters[0].LayoutSlides.Add(SlideLayoutType.Blank, "CustomLayout");
//Add a shape to the LayoutSlide
IShape shape = layoutSlide.Shapes.AddShape(AutoShapeType.Diamond, 30, 20,
400, 300);
//Change the background color for LayoutSlide
layoutSlide.Background.Fill.SolidFill.Color = ColorObject.FromArgb(78, 89,
90);
//Save the PowerPoint file
pptxDoc.Save("LayoutSlide.pptx");
//Close the Presentation instance
```

```
pptxDoc.Close();
```

VB.NET

```
'Creates a PowerPoint instance
Dim pptxDoc As IPresentation = Presentation.Create()
'Add a new LayoutSlide to the PowerPoint file
Dim layoutSlide As ILayoutSlide =
pptxDoc.Masters(0).LayoutSlides.Add(SlideLayoutType.Blank, "CustomLayout")
'Add a shape to the LayoutSlide
Dim shape As IShape = layoutSlide.Shapes.AddShape(AutoShapeType.Diamond, 30,
20, 400, 300)
'Change the background color for LayoutSlide
layoutSlide.Background.Fill.SolidFill.Color = ColorObject.FromArgb(78, 89,
90)
'Save the PowerPoint file
pptxDoc.Save("LayoutSlide.pptx")
'Close the Presentation instance
pptxDoc.Close()
```

Working with Paragraph

Adding Paragraph to slide

All the textual contents in a Presentation document is represented by Paragraphs. You can have any number of paragraphs within a TextBody of a textbox or shape in a PowerPoint presentation.

The following code example demonstrates how to add a paragraph in a slide.

C#

```
//Creates PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the PowerPoint
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Saves the Presentation
pptxDoc.Save("Output.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Creates PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
```

```

'Adds slide to the PowerPoint
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds textbox to the slide
Dim textboxShape As IShape = slide.AddTextBox(0, 0, 500, 500)
'Adds paragraph to the textbody of textbox
Dim paragraph As IParagraph = textboxShape.TextBody.AddParagraph()
'Adds a TextPart to the paragraph
Dim textPart As ITextPart = paragraph.AddTextPart()
'Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base."
'Saves the Presentation
pptxDoc.Save("Output.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Creates PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the PowerPoint
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the PowerPoint
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);

```

```
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```
//Creates PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the PowerPoint
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Applying Paragraph formatting

Each paragraph in a slide can has its own formatting types such as alignment, indent etc. The following code example demonstrates how to format a paragraph in PowerPoint presentation.

C#

```
//Loads the PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Gets the slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets the shape in slide
IShape textboxShape = slide.Shapes[0] as IShape;
//Gets instance of a paragraph in a textbox
IParagraph paragraph = textboxShape.TextBody.Paragraphs[0];
//Applies the first line indent of the paragraph
paragraph.FirstLineIndent = 10;
//Applies the horizontal alignment of the paragraph to center.
paragraph.HorizontalAlignment = HorizontalAlignmentType.Left;
//Applies the left indent of the paragraph
paragraph.LeftIndent = 8;
//Saves the Presentation
pptxDoc.Save("Output.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Loads the PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Gets the slide from Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Gets the shape in slide
Dim textboxShape As IShape = TryCast(slide.Shapes(0), IShape)
'Gets instance of a paragraph in a textbox
Dim paragraph As IParagraph = textboxShape.TextBody.Paragraphs(0)
'Applies the first line indent of the paragraph
paragraph.FirstLineIndent = 10
'Applies the horizontal alignment of the paragraph to center.
paragraph.HorizontalAlignment = HorizontalAlignmentType.Left
'Applies the left indent of the paragraph
paragraph.LeftIndent = 8
'Saves the Presentation
pptxDoc.Save("Output.pptx")
'Closes the Presentation
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
```

```

openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Gets the slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets the shape in slide
IShape textboxShape = slide.Shapes[0] as IShape;
//Gets instance of a paragraph in a textbox
IParagraph paragraph = textboxShape.TextBody.Paragraphs[0];
//Applies the first line indent of the paragraph
paragraph.FirstLineIndent = 10;
//Applies the horizontal alignment of the paragraph to center.
paragraph.HorizontalAlignment = HorizontalAlignmentType.Left;
//Applies the left indent of the paragraph
paragraph.LeftIndent = 8;
//Adds text to paragraph
paragraph.Text = "PowerPoint Presentation";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets the shape in slide
IShape textboxShape = slide.Shapes[0] as IShape;
//Gets instance of a paragraph in a textbox
IParagraph paragraph = textboxShape.TextBody.Paragraphs[0];
//Applies the first line indent of the paragraph
paragraph.FirstLineIndent = 10;
//Applies the horizontal alignment of the paragraph to center.
paragraph.HorizontalAlignment = HorizontalAlignmentType.Left;
//Applies the left indent of the paragraph
paragraph.LeftIndent = 8;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

//"App" is the class of Portable project.

```

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets the shape in slide
IShape textboxShape = slide.Shapes[0] as IShape;
//Gets instance of a paragraph in a textbox
IParagraph paragraph = textboxShape.TextBody.Paragraphs[0];
//Applies the first line indent of the paragraph
paragraph.FirstLineIndent = 10;
//Applies the horizontal alignment of the paragraph to center.
paragraph.HorizontalAlignment = HorizontalAlignmentType.Left;
//Applies the left indent of the paragraph
paragraph.LeftIndent = 8;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);

```

Working with text

With Essential Presentation, you can add or modify the text in a Presentation. Within the paragraph, textual contents are grouped into one or more child elements as TextParts. Each TextPart represents a region of text with a common set of formatted text. The following code example demonstrates how to add text with different formatting into a single paragraph.

C#

```

//Creates the PowerPoint Presentation instance
IPresentation pptxDoc = Presentation.Create();
//Gets the slide from Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds textbox to the slide
IShape textboxShape2 = slide.AddTextBox(500, 0, 400, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph2 = textboxShape2.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPartFormatting = paragraph2.AddTextPart();

```



```

//Adds text to the TextPart
textPartFormatting.Text = "In 2000, AdventureWorks Cycles bought a small
manufacturing plant, located in Mexico. The plant manufactures several
critical subcomponents for the AdventureWorks Cycles product line. These
subcomponents are shipped to the another location for final product
assembly. In 2001, the plant, became the sole manufacturer and distributor
of the touring bicycle product group.";
//Sets the underline color
textPartFormatting.UnderlineColor.SystemColor = Color.Black;
//Retrieves the existing font for modification
IFont font = textPartFormatting.Font;
//Sets the underline type
font.Underline = TextUnderlineType.Single;
//Sets the font weight
font.Bold = true;
//Adds a TextPart to the paragraph
ITextPart textPartFormatting2 = paragraph2.AddTextPart();
//Adds text to the TextPart
textPartFormatting2.Text = "In 2000, AdventureWorks Cycles bought a small
manufacturing plant, located in Mexico.";
//Retrieves the existing font for modification
IFont font2 = textPartFormatting2.Font;
//Sets the font color
font2.Color.SystemColor = Color.Blue;
//Sets the underline type
font2.Underline = TextUnderlineType.WavyDouble;
//Saves the Presentation
pptxDoc.Save("Output.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Loads the PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Gets the slide from Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds textbox to the slide
Dim textboxShape2 As IShape = slide.AddTextBox(500, 0, 400, 500)
'Adds paragraph to the textbox of textbox
Dim paragraph2 As IParagraph = textboxShape2.TextBody.AddParagraph()
'Adds a TextPart to the paragraph
Dim textPartFormatting As ITextPart = paragraph2.AddTextPart()
'Adds text to the TextPart
textPartFormatting.Text = "In 2000, AdventureWorks Cycles bought a small
manufacturing plant, located in Mexico. It manufactures several critical
subcomponents for the AdventureWorks Cycles product line. These
subcomponents are shipped to the another location for final product
assembly. In 2001, the plant, became the sole manufacturer and distributor
of the touring bicycle product group."
'Sets the underline color
textPartFormatting.UnderlineColor.SystemColor = Color.Black
'Retrieves the existing font for modification
Dim font As IFont = textPartFormatting.Font
'Sets the underline type
font.Underline = TextUnderlineType.[Single]

```

```

'Sets the font weight
font.Bold = True
'Adds a TextPart to the paragraph
Dim textPartFormatting2 As ITextPart = paragraph2.AddTextPart()
'Adds text to the TextPart
textPartFormatting2.Text = "In 2000, AdventureWorks Cycles bought a small
manufacturing plant, located in Mexico."
'Retrieves the existing font for modification
Dim font2 As IFont = textPartFormatting2.Font
'Sets the font color
font2.Color.SystemColor = Color.Blue
'Sets the underline type
font2.Underline = TextUnderlineType.WavyDouble
'Saves the Presentation
pptxDoc.Save("Output.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Create the PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Gets the slide from Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds textbox to the slide
IShape textboxShape2 = slide.AddTextBox(500, 0, 400, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph2 = textboxShape2.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPartFormatting = paragraph2.AddTextPart();
//Adds text to the TextPart
textPartFormatting.Text = "In 2000, AdventureWorks Cycles bought a small
manufacturing plant, located in Mexico. The plant manufactures several
critical subcomponents for the AdventureWorks Cycles product line. These
subcomponents are shipped to the another location for final product
assembly. In 2001, the plant, became the sole manufacturer and distributor
of the touring bicycle product group.";
//Sets the underline color
textPartFormatting.UnderlineColor = ColorObject.Black;
//Retrieves the existing font for modification
IFont font = textPartFormatting.Font;
//Sets the underline type
font.Underline = TextUnderlineType.Single;
//Sets the font weight
font.Bold = true;
//Adds a TextPart to the paragraph
ITextPart textPartFormatting2 = paragraph2.AddTextPart();
//Adds text to the TextPart
textPartFormatting2.Text = "In 2000, AdventureWorks Cycles bought a small
manufacturing plant, located in Mexico.";
//Retrieves the existing font for modification
IFont font2 = textPartFormatting2.Font;
//Sets the font color
font2.Color = ColorObject.Blue;
//Sets the underline type
font2.Underline = TextUnderlineType.WavyDouble;

```

```
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Creates the PowerPoint Presentation instance
IPresentation pptxDoc = Presentation.Create();
//Adds new slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds textbox to the slide
IShape textboxShape2 = slide.AddTextBox(500, 0, 400, 500);
//Adds paragraph to the textbox of textbox
IParagraph paragraph2 = textboxShape2.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPartFormatting = paragraph2.AddTextPart();
//Adds text to the TextPart
textPartFormatting.Text = "In 2000, AdventureWorks Cycles bought a small  
manufacturing plant, located in Mexico. The plant manufactures several  
critical subcomponents for the AdventureWorks Cycles product line. These  
subcomponents are shipped to the another location for final product  
assembly. In 2001, the plant, became the sole manufacturer and distributor  
of the touring bicycle product group.";
//Sets the underline color
textPartFormatting.UnderlineColor = ColorObject.AliceBlue;
//Retrieves the existing font for modification
IFont font = textPartFormatting.Font;
//Sets the underline type
font.Underline = TextUnderlineType.Single;
//Sets the font weight
font.Bold = true;
//Adds a TextPart to the paragraph
ITextPart textPartFormatting2 = paragraph2.AddTextPart();
//Adds text to the TextPart
textPartFormatting2.Text = "In 2000, AdventureWorks Cycles bought a small  
manufacturing plant, located in Mexico.";
//Retrieves the existing font for modification
IFont font2 = textPartFormatting2.Font;
//Sets the font color
font2.Color = ColorObject.BlanchedAlmond;
//Sets the underline type
font2.Underline = TextUnderlineType.WavyDouble;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```

//Creates the PowerPoint Presentation instance
IPresentation pptxDoc = Presentation.Create();
//Adds new slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds textbox to the slide
IShape textboxShape2 = slide.AddTextBox(500, 0, 400, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph2 = textboxShape2.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPartFormatting = paragraph2.AddTextPart();
//Adds text to the TextPart
textPartFormatting.Text = "In 2000, AdventureWorks Cycles bought a small
manufacturing plant, located in Mexico. The plant manufactures several
critical subcomponents for the AdventureWorks Cycles product line. These
subcomponents are shipped to the another location for final product
assembly. In 2001, the plant, became the sole manufacturer and distributor
of the touring bicycle product group.";
//Sets the underline color
textPartFormatting.UnderlineColor = ColorObject.AliceBlue;
//Retrieves the existing font for modification
IFont font = textPartFormatting.Font;
//Sets the underline type
font.Underline = TextUnderlineType.Single;
//Sets the font weight
font.Bold = true;
//Adds a TextPart to the paragraph
ITextPart textPartFormatting2 = paragraph2.AddTextPart();
//Adds text to the TextPart
textPartFormatting2.Text = "In 2000, AdventureWorks Cycles bought a small
manufacturing plant, located in Mexico.";
//Retrieves the existing font for modification
IFont font2 = textPartFormatting2.Font;
//Sets the font color
font2.Color = ColorObject.BlanchedAlmond;
//Sets the underline type
font2.Underline = TextUnderlineType.WavyDouble;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Modifying text

You can modify a text by accessing the existing paragraphs in a Presentation. The following code example demonstrates how to modify the content in a paragraph.

C#

```
//Opens an existing Presentation from file system.
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape.
IShape shape = slide.Shapes[0] as IShape;
//Retrieves the first paragraph of the shape.
IParagraph paragraph = shape.TextBody.Paragraphs[0];
//Retrieves the first TextPart of the shape.
ITextPart textPart = paragraph.TextParts[0];
//Modifies the text content of the TextPart.
textPart.Text = "Hello Presentation";
//Saves the presentation to the file system.
pptxDoc.Save("Result.pptx");
//Closes the Presentation.
pptxDoc.Close();
```

VB.NET

```
'Opens an existing Presentation from file system.
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Retrieves the first slide from Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Retrieves the first shape.
Dim shape As IShape = TryCast(slide.Shapes(0), IShape)
'Retrieves the first paragraph of the shape.
Dim paragraph As IParagraph = shape.TextBody.Paragraphs(0)
'Retrieves the first TextPart of the shape.
Dim textPart As ITextPart = paragraph.TextParts(0)
'Modifies the text content of the TextPart.
textPart.Text = "Hello Presentation"
'Saves the presentation to the file system.
pptxDoc.Save("Result.pptx")
'Closes the Presentation.
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
```

```

//Retrieves the first shape.
IShape shape = slide.Shapes[0] as IShape;
//Retrieves the first paragraph of the shape.
IParagraph paragraph = shape.TextBody.Paragraphs[0];
//Retrieves the first TextPart of the shape.
ITextPart textPart = paragraph.TextParts[0];
//Modifies the text content of the TextPart.
textPart.Text = "Hello Presentation";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Result";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape.
IShape shape = slide.Shapes[0] as IShape;
//Retrieves the first paragraph of the shape.
IParagraph paragraph = shape.TextBody.Paragraphs[0];
//Retrieves the first TextPart of the shape.
ITextPart textPart = paragraph.TextParts[0];
//Modifies the text content of the TextPart.
textPart.Text = "Hello Presentation";
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation.
pptxDoc.Close();

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape.
IShape shape = slide.Shapes[0] as IShape;
//Retrieves the first paragraph of the shape.
IParagraph paragraph = shape.TextBody.Paragraphs[0];
//Retrieves the first TextPart of the shape.

```

```

ITextPart textPart = paragraph.TextParts[0];
//Modifies the text content of the TextPart.
textPart.Text = "Hello Presentation";
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Enabling shrink text on overflow option

In a PowerPoint slide, if you add a text more than a shape can hold, the text will overflow from the shape. But by using a Shrink text on overflow option, you can fit a large text within a shape. The following code example demonstrates how to enable this property.

C#

```

// Create a new PowerPoint file.
using (IPresentation ppDoc = Presentation.Create())
{
    // Add a slide to the PowerPoint file.
    ISlide slide = ppDoc.Slides.Add(SlideLayoutType.Blank);
    // Add a text box to the slide
    IShape textBox = slide.Shapes.AddTextBox(100, 100, 100, 100);
    //Add text to the text box.
    textBox.TextBody.AddParagraph("AdventureWorks Cycles, the fictitious company
on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.");
    //Set the property to shrink text on overflow.
    textBox.TextBody.FitTextOption = FitTextOption.ShrinkTextOnOverflow;
    // Save the PowerPoint file
    ppDoc.Save("Sample.pptx");
}

```

VB.NET

```

'Create a new PowerPoint file
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds slide to the PowerPoint
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds textbox to the slide
Dim textboxShape As IShape = slide.AddTextBox(0, 0, 500, 500)

```

```

'Adds paragraph to the textbody of textbox
Dim paragraph As IParagraph =
textboxShape.TextBody.AddParagraph("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.")
'Set the property to shrink text on overflow.
textboxShape.TextBody.FitTextOption = FitTextOption.ShrinkTextOnOverflow
'Save the PowerPoint file
pptxDoc.Save("Output.pptx")
'Close the PowerPoint file
pptxDoc.Close()

```

Note: The shrink text on overflow is not supported in UWP, ASP.NET CORE and Xamarin platforms.

Removing the paragraph

The following code example demonstrates how to remove a paragraph from a slide.

C#

```

//Opens an existing Presentation from file system.
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape
IShape shape = slide.Shapes[0] as IShape;
//Retrieves the first paragraph of the shape
IParagraph paragraph = shape.TextBody.Paragraphs[0];
//Removes the first paragraph from the textbody of the shape
shape.TextBody.Paragraphs.Remove(paragraph);
//Saves the presentation to the file system
pptxDoc.Save("Result.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Opens an existing Presentation from file system.
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Retrieves the first slide from Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Retrieves the first shape.
Dim shape As IShape = TryCast(slide.Shapes(0), IShape)
'Retrieves the first paragraph of the shape.
Dim paragraph As IParagraph = shape.TextBody.Paragraphs(0)
'Removes the first paragraph from the textbody of the shape.
shape.TextBody.Paragraphs.Remove(paragraph)
'Saves the presentation to the file system.
pptxDoc.Save("Result.pptx")
'Closes the Presentation.
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();

```



```

openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape.
IShape shape = slide.Shapes[0] as IShape;
//Retrieves the first paragraph of the shape.
IParagraph paragraph = shape.TextBody.Paragraphs[0];
//Removes the first paragraph from the textbody of the shape
shape.TextBody.Paragraphs.Remove(paragraph);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Result";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape
IShape shape = slide.Shapes[0] as IShape;
//Retrieves the first paragraph of the shape
IParagraph paragraph = shape.TextBody.Paragraphs[0];
//Removes the first paragraph from the textbody of the shape
shape.TextBody.Paragraphs.Remove(paragraph);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape

```

```

IShape shape = slide.Shapes[0] as IShape;
//Retrieves the first paragraph of the shape
IParagraph paragraph = shape.TextBody.Paragraphs[0];
//Removes the first paragraph from the textbody of the shape
shape.TextBody.Paragraphs.Remove(paragraph);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Working with lists

Creating a simple list

Essential Presentation allows you to create simple and multi-level lists that make the content easier for reading. In PowerPoint, Presentation lists consists of the following types

1. Numbered list
2. Bulleted list
3. Picture list

Numbered List

The following code example illustrates how to create a numbered list:

C#

```

//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
// Adds a textbox to hold the list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 270);
// Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;

```

```

//Sets the starting value as 1
paragraph.ListFormat.StartValue = 1;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
//Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
//Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located in Washington with 290 employees, several regional sales teams are
located throughout their market base.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
//Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Saves the Presentation to the file system.
pptxDoc.Save("Sample.pptx");
Process.Start("Sample.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Creates a new Presentation instance.
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds the slide into the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds a textbox to hold the list
Dim textBoxShape As IShape = slide.AddTextBox(65, 140, 410, 270)
'Adds a new paragraph with the text in the left hand side textbox.

```

```

Dim paragraph As IParagraph =
textBoxShape.TextBody.AddParagraph("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.")
'Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Numbered
'Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod
'Sets the starting value as 1
paragraph.ListFormat.StartValue = 1
'Sets the list level as 1
paragraph.IndentLevelNumber = 1
'Sets the hanging value
paragraph.FirstLineIndent = -20
'Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
paragraph.ListFormat.Size = 100
'Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.")
'Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered
'Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod
'Sets the list level as 1
paragraph.IndentLevelNumber = 1
'Sets the hanging value
paragraph.FirstLineIndent = -20
'Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
paragraph.ListFormat.Size = 100
'Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located Washington with 290 employees, several regional sales teams are
located throughout their market base.")
'Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered
'Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod
'Sets the list level as 1
paragraph.IndentLevelNumber = 1
'Sets the hanging value
paragraph.FirstLineIndent = -20
'Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
paragraph.ListFormat.Size = 100
'Saves the Presentation to the file system.
pptxDoc.Save("Sample.pptx")
Process.Start("Sample.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```
//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a textbox to hold the list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 270);
//Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the starting value as 1
paragraph.ListFormat.StartValue = 1;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located in Washington with 290 employees, several regional sales teams are
located throughout their market base.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
```

```

paragraph.ListFormat.Size = 100;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
// Adds a textbox to hold the list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 270);
// Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the starting value as 1
paragraph.ListFormat.StartValue = 1;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.

```

```

paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located in Washington with 290 employees, several regional sales teams are
located throughout their market base.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
// Adds a textbox to hold the list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 270);
// Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the starting value as 1
paragraph.ListFormat.StartValue = 1;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;

```

```

//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located in Washington with 290 employees, several regional sales teams are
located throughout their market base.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Bulleted list

The following code example demonstrates how to create a simple bulleted list.

C#

```

//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds the slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
// Adds a textbox to hold the list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 250);
// Adds a new paragraph with the text in the left hand side textbox.

```



```
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks  
Cycles, the fictitious company on which the AdventureWorks sample databases  
are based, is a large, multinational manufacturing company.");  
//Sets the list type as bulleted  
paragraph.ListFormat.Type = ListType.Bulleted;  
//Sets the bullet character for this list  
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);  
//Sets the hanging value  
paragraph.FirstLineIndent = -20;  
//Sets the list level as 1  
paragraph.IndentLevelNumber = 1;  
// Sets the font for the bullet character  
paragraph.ListFormat.FontName = "Symbol";  
// Sets the bullet character size. Here, 100 means 100% of its text.  
Possible values can range from 25 to 400.  
paragraph.ListFormat.Size = 100;  
// Adds another paragraph with the text in the left hand side textbox.  
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and  
sells metal and composite bicycles to North American, European and Asian  
commercial markets.");  
//Sets the list type as bulleted  
paragraph.ListFormat.Type = ListType.Bulleted;  
//Sets the bullet character for this list  
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);  
//Sets the hanging value  
paragraph.FirstLineIndent = -20;  
//Sets the list level as 1  
paragraph.IndentLevelNumber = 1;  
// Sets the font for the bullet character  
paragraph.ListFormat.FontName = "Symbol";  
// Sets the bullet character size. Here, 100 means 100% of its text.  
Possible values can range from 25 to 400.  
paragraph.ListFormat.Size = 100;  
// Adds another paragraph with the text in the left hand side textbox.  
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is  
located in Washington with 290 employees, several regional sales teams are  
located throughout their market base.");  
//Sets the list type as bulleted  
paragraph.ListFormat.Type = ListType.Bulleted;  
//Sets the bullet character for this list  
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);  
//Sets the hanging value  
paragraph.FirstLineIndent = -20;  
//Sets the list level as 1  
paragraph.IndentLevelNumber = 1;  
// Sets the font of the bullet character  
paragraph.ListFormat.FontName = "Symbol";  
// Sets the bullet character size. Here, 100 means 100% of its text.  
Possible values can range from 25 to 400.  
paragraph.ListFormat.Size = 100;  
//Saves the Presentation to the file system.  
pptxDoc.Save("Sample.pptx");  
Process.Start("Sample.pptx");  
//Closes the Presentation  
pptxDoc.Close();
```

VB.NET

```
'Creates a new Presentation instance.
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds the slide into the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
' Adds a textbox to hold the list
Dim textBoxShape As IShape = slide.AddTextBox(65, 140, 410, 250)
' Adds a new paragraph with the text in the left hand side textbox.
Dim paragraph As IParagraph =
textBoxShape.TextBody.AddParagraph("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.")
'Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Bulleted
'Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183)
'Sets the hanging value
paragraph.FirstLineIndent = -20
'Sets the list level as 1
paragraph.IndentLevelNumber = 1
' Sets the font for the bullet character
paragraph.ListFormat.FontName = "Symbol"
' Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
paragraph.ListFormat.Size = 100
' Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.")
'Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Bulleted
'Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183)
'Sets the hanging value
paragraph.FirstLineIndent = -20
'Sets the list level as 1
paragraph.IndentLevelNumber = 1
' Sets the font for the bullet character
paragraph.ListFormat.FontName = "Symbol"
' Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
paragraph.ListFormat.Size = 100
' Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located in Washington with 290 employees, several regional sales teams are
located throughout their market base.")
'Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Bulleted
'Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183)
'Sets the hanging value
paragraph.FirstLineIndent = -20
'Sets the list level as 1
paragraph.IndentLevelNumber = 1
' Sets the font of the bullet character
paragraph.ListFormat.FontName = "Symbol"
```

```
' Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
paragraph.ListFormat.Size = 100
'Saves the Presentation to the file system.
pptxDoc.Save("Sample.pptx")
Process.Start("Sample.pptx")
'Closes the Presentation
pptxDoc.Close()
```

UWP

```
//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds the slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a textbox to hold the list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 250);
//Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
//Sets the font for the bullet character
paragraph.ListFormat.FontName = "Symbol";
//Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
//Sets the font for the bullet character
paragraph.ListFormat.FontName = "Symbol";
//Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located in Washington with 290 employees, several regional sales teams are
located throughout their market base.");
//Sets the list type as bulleted
```

```

paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
//Sets the font of the bullet character
paragraph.ListFormat.FontName = "Symbol";
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds the slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
// Adds a textbox to hold the list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 250);
// Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the font for the bullet character
paragraph.ListFormat.FontName = "Symbol";
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);

```

```
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the font for the bullet character
paragraph.ListFormat.FontName = "Symbol";
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located in Washington with 290 employees, several regional sales teams are
located throughout their market base.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the font of the bullet character
paragraph.ListFormat.FontName = "Symbol";
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```
//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds the slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
// Adds a textbox to hold the list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 250);
// Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the font for the bullet character
paragraph.ListFormat.FontName = "Symbol";
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
```

```
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the font for the bullet character
paragraph.ListFormat.FontName = "Symbol";
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located in Washington with 290 employees, several regional sales teams are
located throughout their market base.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Bulleted;
//Sets the bullet character for this list
paragraph.ListFormat.BulletCharacter = Convert.ToChar(183);
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the font of the bullet character
paragraph.ListFormat.FontName = "Symbol";
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Picture List

The following code example demonstrates how to create a simple picture list.

C#

```
//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds the slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
// Adds a textbox to hold the list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 270);
// Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Picture;
//Sets the image for the list.
paragraph.ListFormat.Picture(new
MemoryStream(Syncfusion.Drawing.Image.FromFile("Image.png").ImageData));
// Sets the picture size. Here, 100 means 100% of its text. Possible values
can range from 25 to 400.
paragraph.ListFormat.Size = 150;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as picture
paragraph.ListFormat.Type = ListType.Picture;
//Sets the image for the list.
paragraph.ListFormat.Picture(new
MemoryStream(Syncfusion.Drawing.Image.FromFile("Image.png").ImageData));
// Sets the picture size. Here, 100 means 100% of its text. Possible values
can range from 25 to 400.
paragraph.ListFormat.Size = 150;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Saves the Presentation to the file system.
pptxDoc.Save("Sample.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Creates a new Presentation instance.
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds the slide into the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
' Adds a textbox to hold the list
Dim textBoxShape As IShape = slide.AddTextBox(65, 140, 410, 270)
```

```

' Adds a new paragraph with the text in the left hand side textbox.
Dim paragraph As IParagraph =
textBoxShape.TextBody.AddParagraph("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.")
'Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Picture
'Sets the image for the list.
paragraph.ListFormat.Picture(New MemoryStream (Syncfusion.Drawing.
Image.FromFile("Image.png").ImageData))
' Sets the picture size. Here, 100 means 100% of its text. Possible values
can range from 25 to 400.
paragraph.ListFormat.Size = 150
'Sets the list level as 1
paragraph.IndentLevelNumber = 1
' Sets the hanging value
paragraph.FirstLineIndent = -20
' Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.")
'Sets the list type as picture
paragraph.ListFormat.Type = ListType.Picture
'Sets the image for the list.
paragraph.ListFormat.Picture(New MemoryStream (Syncfusion.Drawing.
Image.FromFile("Image.png").ImageData))
' Sets the picture size. Here, 100 means 100% of its text. Possible values
can range from 25 to 400.
paragraph.ListFormat.Size = 150
'Sets the list level as 1
paragraph.IndentLevelNumber = 1
' Sets the hanging value
paragraph.FirstLineIndent = -20
'Saves the Presentation to the file system.
pptxDoc.Save("Sample.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds the slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a textbox to hold the list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 270);
//Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Picture;
//Sets the image for the list.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream pictureStream =
assembly.GetManifestResourceStream("UWP.Data.Image.png");

```



```

paragraph.ListFormat.Picture(pictureStream);
//Sets the picture size. Here, 100 means 100% of its text. Possible values
can range from 25 to 400.
paragraph.ListFormat.Size = 150;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as picture
paragraph.ListFormat.Type = ListType.Picture;
//Sets the image for the list.
paragraph.ListFormat.Picture(new
MemoryStream(Syncfusion.Drawing.Image.FromFile("Image.png").ImageData));
//Sets the picture size. Here, 100 means 100% of its text. Possible values
can range from 25 to 400.
paragraph.ListFormat.Size = 150;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
//Sets the hanging value
paragraph.FirstLineIndent = -20;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds the slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
// Adds a textbox to hold the list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 270);
// Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Picture;
//Sets the image for the list.
paragraph.ListFormat.Picture(pictureStream);
// Sets the picture size. Here, 100 means 100% of its text. Possible values
can range from 25 to 400.
paragraph.ListFormat.Size = 150;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;

```

```
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as picture
paragraph.ListFormat.Type = ListType.Picture;
//Sets the image for the list.
paragraph.ListFormat.Picture(pictureStream);
// Sets the picture size. Here, 100 means 100% of its text. Possible values
can range from 25 to 400.
paragraph.ListFormat.Size = 150;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```
//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds the slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
// Adds a textbox to hold the list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 270);
// Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Picture;
//Sets the image for the list.
paragraph.ListFormat.Picture(pictureStream);
// Sets the picture size. Here, 100 means 100% of its text. Possible values
can range from 25 to 400.
paragraph.ListFormat.Size = 150;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as picture
paragraph.ListFormat.Type = ListType.Picture;
//Sets the image for the list.
paragraph.ListFormat.Picture(pictureStream);
// Sets the picture size. Here, 100 means 100% of its text. Possible values
can range from 25 to 400.
```

```

paragraph.ListFormat.Size = 150;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Creating a Multilevel List

You can create a multi-level list by setting the indentation levels. By default, the level begins at 0 and increments by 1 for each level. A list can be incremented or decremented from levels 0 to 8 as like Microsoft PowerPoint.

The following code example demonstrates how to create a multilevel list.

C#

```

//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds the slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a textbox to hold the bulleted list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 250);
//Adds paragraph to the textbox
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the starting value as 1
paragraph.ListFormat.StartValue = 1;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Adds paragraph to the textbox

```

```

paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as lower case alphabet following
by period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.AlphaLcPeriod;
//Sets the list level as 2
paragraph.IndentLevelNumber = 2;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Add paragraph to the textbox
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located Washington with 290 employees, several regional sales teams are
located throughout their market base.");
//Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as roman number lower casing
following by period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.RomanLcPeriod;
//Sets the list level as 3
paragraph.IndentLevelNumber = 3;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Adds paragraph to the textbox
paragraph = textBoxShape.TextBody.AddParagraph("These subcomponents are
shipped to another location for final product assembly");
//Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered;
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Saves the Presentation to the file system.
pptxDoc.Save("MultiLevelList.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Creates a new Presentation instance.
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds the slide into the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds a textbox to hold the bulleted list
Dim textBoxShape As IShape = slide.AddTextBox(65, 140, 410, 250)
'Adds paragraph to the textbox
Dim paragraph As IParagraph =
textBoxShape.TextBody.AddParagraph("AdventureWorks Cycles, the fictitious
company on which the AdventureWorks sample databases are based, is a large,
multinational manufacturing company.")
'Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered

```

```

'Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod
'Sets the starting value as 1
paragraph.ListFormat.StartValue = 1
'Sets the list level as 1
paragraph.IndentLevelNumber = 1
' Sets the hanging value
paragraph.FirstLineIndent = -20
'Adds paragraph to the textbox
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.")
'Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered
'Sets the numbered style (list numbering) as lower case alphabet following
by period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.AlphaLcPeriod
'Sets the list level as 2
paragraph.IndentLevelNumber = 2
' Sets the hanging value
paragraph.FirstLineIndent = -20
'Adds paragraph to the textbox
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located Washington with 290 employees, several regional sales teams are
located throughout their market base.")
'Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered
'Sets the numbered style (list numbering) as roman number lower casing
following by period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.RomanLcPeriod
'Sets the list level as 3
paragraph.IndentLevelNumber = 3
' Sets the hanging value
paragraph.FirstLineIndent = -20
'Adds paragraph to the textbox
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located in Washington with 290 employees, several regional sales teams are
located throughout their market base.")
'Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod
'Sets the list level as 1
paragraph.IndentLevelNumber = 1
' Sets the hanging value
paragraph.FirstLineIndent = -20
'Saves the Presentation to the file system.
pptxDoc.Save("MultiLevelList.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds the slide into the Presentation

```

```
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);  
//Adds a textbox to hold the bulleted list  
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 250);  
//Adds paragraph to the textbox  
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks  
Cycles, the fictitious company on which the AdventureWorks sample databases  
are based, is a large, multinational manufacturing company.");  
//Sets the list type as Numbered list  
paragraph.ListFormat.Type = ListType.Numbered;  
//Sets the numbered style (list numbering) as Arabic number following by  
period.  
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;  
//Sets the starting value as 1  
paragraph.ListFormat.StartValue = 1;  
//Sets the list level as 1  
paragraph.IndentLevelNumber = 1;  
// Sets the hanging value  
paragraph.FirstLineIndent = -20;  
//Adds paragraph to the textbox  
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and  
sells metal and composite bicycles to North American, European and Asian  
commercial markets.");  
//Sets the list type as Numbered list  
paragraph.ListFormat.Type = ListType.Numbered;  
//Sets the numbered style (list numbering) as lower case alphabet following  
by period.  
paragraph.ListFormat.NumberStyle = NumberedListStyle.AlphaLcPeriod;  
//Sets the list level as 2  
paragraph.IndentLevelNumber = 2;  
// Sets the hanging value  
paragraph.FirstLineIndent = -20;  
//Add paragraph to the textbox  
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is  
located Washington with 290 employees, several regional sales teams are  
located throughout their market base.");  
//Sets the list type as Numbered list  
paragraph.ListFormat.Type = ListType.Numbered;  
//Sets the numbered style (list numbering) as roman number lower casing  
following by period.  
paragraph.ListFormat.NumberStyle = NumberedListStyle.RomanLcPeriod;  
//Sets the list level as 3  
paragraph.IndentLevelNumber = 3;  
// Sets the hanging value  
paragraph.FirstLineIndent = -20;  
//Adds paragraph to the textbox  
paragraph = textBoxShape.TextBody.AddParagraph("These subcomponents are  
shipped to another location for final product assembly");  
//Sets the list type as Numbered list  
paragraph.ListFormat.Type = ListType.Numbered;  
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;  
//Sets the list level as 1  
paragraph.IndentLevelNumber = 1;  
// Sets the hanging value  
paragraph.FirstLineIndent = -20;  
//Initializes FileSavePicker  
FileSavePicker savePicker = new FileSavePicker();  
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
```

```

savePicker.SuggestedFileName = "MultiLevelList";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds the slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a textbox to hold the bulleted list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 250);
//Adds paragraph to the textbox
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the starting value as 1
paragraph.ListFormat.StartValue = 1;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Adds paragraph to the textbox
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as lower case alphabet following
by period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.AlphaLcPeriod;
//Sets the list level as 2
paragraph.IndentLevelNumber = 2;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Add paragraph to the textbox
paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located Washington with 290 employees, several regional sales teams are
located throughout their market base.");
//Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as roman number lower casing
following by period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.RomanLcPeriod;
//Sets the list level as 3
paragraph.IndentLevelNumber = 3;
// Sets the hanging value

```

```

paragraph.FirstLineIndent = -20;
//Adds paragraph to the textbox
paragraph = textBoxShape.TextBody.AddParagraph("These subcomponents are
shipped to another location for final product assembly");
//Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered;
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("MultiLevelList.pptx",
FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

//Creates a new Presentation instance.
IPresentation pptxDoc = Presentation.Create();
//Adds the slide into the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a textbox to hold the bulleted list
IShape textBoxShape = slide.AddTextBox(65, 140, 410, 250);
//Adds paragraph to the textbox
IParagraph paragraph = textBoxShape.TextBody.AddParagraph("AdventureWorks
Cycles, the fictitious company on which the AdventureWorks sample databases
are based, is a large, multinational manufacturing company.");
//Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the starting value as 1
paragraph.ListFormat.StartValue = 1;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Adds paragraph to the textbox
paragraph = textBoxShape.TextBody.AddParagraph("The company manufactures and
sells metal and composite bicycles to North American, European and Asian
commercial markets.");
//Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as lower case alphabet following
by period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.AlphaLcPeriod;
//Sets the list level as 2
paragraph.IndentLevelNumber = 2;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Add paragraph to the textbox

```



```

paragraph = textBoxShape.TextBody.AddParagraph("While its base operation is
located Washington with 290 employees, several regional sales teams are
located throughout their market base.");
//Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as roman number lower casing
following by period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.RomanLcPeriod;
//Sets the list level as 3
paragraph.IndentLevelNumber = 3;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Adds paragraph to the textbox
paragraph = textBoxShape.TextBody.AddParagraph("These subcomponents are
shipped to another location for final product assembly");
//Sets the list type as Numbered list
paragraph.ListFormat.Type = ListType.Numbered;
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("MultiLevelLi
st.pptx", "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("MultiLevelList.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

The above code example generates a multi-level list in Presentation as follows.

1. Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company.
 - a. The company manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets.
 - i. While its base operation is located in Bothell, Washington with 290 employees, several regional sales teams are located throughout their market base.
2. These subcomponents are shipped to the Bothell location for final product assembly

Working with Shapes

Adding shapes to a slide

In every slide, there is a shape collection that can contain any form of graphical objects such as AutoShape, chart, text, or picture. You can add any shape element to this collection. The IShape is the base type for the shape elements.

The following code example demonstrates how to add an AutoShape and image to the shape collection of a slide.

C#

```
//Creates an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds normal shape to slide
slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300);
//Creates an instance for image as stream
Stream imageStream = File.Open("Image.jpg", FileMode.Open);
//Add picture to the shape collection
IPicture picture = slide.Shapes.AddPicture(imageStream, 373, 83, 526, 382);
//Saves the Presentation
pptxDoc.Save("Sample.pptx");
//Closes the stream
imageStream.Close();
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Creates an instance for PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds a blank slide to Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
```

```

'Adds normal shape to slide
slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300)
'Creates an instance for image as stream
Dim imageStream As Stream = File.Open("Image.jpg", FileMode.Open)
'Adds picture to the shape collection
Dim picture As IPicture = slide.Shapes.AddPicture(imageStream, 373, 83, 500,
382)
'Saves the Presentation
pptxDoc.Save("Sample.pptx")
'Closes the stream
imageStream.Close()
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Creates an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds normal shape to slide
slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300);
//Creates an instance for image as stream
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream pictureStream =
assembly.GetManifestResourceStream("UWP.Data.Image.jpg");
//Add picture to the shape collection
IPicture picture = slide.Shapes.AddPicture(pictureStream, 373, 83, 526,
382);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds normal shape to slide
slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300);
//Creates an instance for image as stream
FileStream imageStream = new FileStream(imagePath, FileMode.Open);
//Add picture to the shape collection
IPicture picture = slide.Shapes.AddPicture(imageStream, 373, 83, 526, 382);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the stream

```

```
imageStream.Close();
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```
//Creates an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds normal shape to slide
slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300);
//Creates an instance for image as stream
Stream imageStream = assembly.GetManifestResourceStream(picturePath);
//Add picture to the shape collection
IPicture picture = slide.Shapes.AddPicture(imageStream, 373, 83, 526, 382);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
//Closes the stream
imageStream.Close();
```

Iterating through shapes

You can iterate through the shapes in a PowerPoint slide. The following code example demonstrates how to iterate through the shapes present in a slide for modifying its properties.

C#

```
//Opens an existing Presentation from the file system
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Iterates through shapes in a slide and sets title
foreach (IShape shape in pptxDoc.Slides[0].Shapes)
{
    if (shape is IPicture)
        shape.Title = "Picture";
    else if (shape is IShape)
        shape.Title = "AutoShape";
}
//Saves the Presentation
```

```
pptxDoc.Save("Output.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Opens an existing Presentation from the file system
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Iterates through shapes in a slide and sets title
For Each shape As IShape In pptxDoc.Slides(0).Shapes
If TypeOf shape Is IPicture Then
shape.Title = "Picture"
ElseIf TypeOf shape Is IShape Then
shape.Title = "AutoShape"
End If
Next
'Saves the Presentation
pptxDoc.Save("Output.pptx")
'Closes the Presentation
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Iterates through shapes in a slide and sets title
foreach(IShape shape in pptxDoc.Slides[0].Shapes)
{
if (shape is IPicture)
shape.Title = "Picture";
else if (shape is IShape)
shape.Title = "AutoShape";
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
```

```
//Iterates through shapes in a slide and sets title
foreach(ISHape shape in pptxDoc.Slides[0].Shapes)
{
    if (shape is IPicture)
        shape.Title = "Picture";
    else if (shape is IShape)
        shape.Title = "AutoShape";
}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Iterates through shapes in a slide and sets title
foreach(ISHape shape in pptxDoc.Slides[0].Shapes)
{
    if (shape is IPicture)
        shape.Title = "Picture";
    else if (shape is IShape)
        shape.Title = "AutoShape";
}
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Specifying shape properties

The shape properties can be used to format and modify the shapes in a slide. The following code example demonstrates how to apply formatting to a shape.

C#

```

//Creates instance for PowerPoint
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Gets the first slide of the Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets the shape of the slide
IShape shape = slide.Shapes[0] as IShape;
//Sets the shape name.
shape.ShapeName = "Shape1";
//Retrieves the line format of the shape.
ILineFormat lineFormat = shape.LineFormat;
//Sets the dash style of the line format.
lineFormat.DashStyle = LineDashStyle.DashDotDot;
//Sets the weight of the line format.
lineFormat.Weight = 3;
//Sets the pattern fill type to shape
shape.Fill.FillType = FillType.Pattern;
//Chooses the type of pattern
shape.Fill.PatternFill.Pattern = PatternFillType.DashedDownwardDiagonal;
//Sets the fore color
shape.Fill.PatternFill.ForeColor = ColorObject.AliceBlue;
//Sets the back color
shape.Fill.PatternFill.BackColor = ColorObject.DarkSalmon;
//Saves the Presentation
pptxDoc.Save("Output.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Creates instance for PowerPoint
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Gets the first slide of the Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Gets the shape of the slide
Dim shape As IShape = TryCast(slide.Shapes(0), IShape)
'Sets the shape name.
shape.ShapeName = "Shape1"
'Retrieves the line format of the shape.
Dim lineFormat As ILineFormat = shape.LineFormat
'Sets the dash style of the line format.
lineFormat.DashStyle = LineDashStyle.DashDotDot
'Sets the weight of the line format.
lineFormat.Weight = 3
'Sets the pattern fill type to shape
shape.Fill.FillType = FillType.Pattern
'Chooses the type of pattern
shape.Fill.PatternFill.Pattern = PatternFillType.DashedDownwardDiagonal
'Sets the fore color
shape.Fill.PatternFill.ForeColor = ColorObject.AliceBlue
'Sets the back color
shape.Fill.PatternFill.BackColor = ColorObject.DarkSalmon
'Saves the Presentation
pptxDoc.Save("Output.pptx")
'Closes the Presentation

```

```
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Gets the first slide of the Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets the shape of the slide
IShape shape = slide.Shapes[0] as IShape;
//Sets the shape name.
shape.ShapeName = "Shape1";
//Retrieves the line format of the shape.
ILineFormat lineFormat = shape.LineFormat;
//Sets the dash style of the line format.
lineFormat.DashStyle = LineDashStyle.DashDotDot;
//Sets the weight of the line format.
lineFormat.Weight = 3;
//Sets the pattern fill type to shape
shape.Fill.FillType = FillType.Pattern;
//Chooses the type of pattern
shape.Fill.PatternFill.Pattern = PatternFillType.DashedDownwardDiagonal;
//Sets the fore color
shape.Fill.PatternFill.ForeColor = ColorObject.AliceBlue;
//Sets the back color
shape.Fill.PatternFill.BackColor = ColorObject.DarkSalmon;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide of the Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets the shape of the slide
IShape shape = slide.Shapes[0] as IShape;
//Sets the shape name.
shape.ShapeName = "Shape1";
//Retrieves the line format of the shape.
ILineFormat lineFormat = shape.LineFormat;
```



```
//Sets the dash style of the line format.
lineFormat.DashStyle = LineDashStyle.DashDotDot;
//Sets the weight of the line format.
lineFormat.Weight = 3;
//Sets the pattern fill type to shape
shape.Fill.FillType = FillType.Pattern;
//Chooses the type of pattern
shape.Fill.PatternFill.Pattern = PatternFillType.DashedDownwardDiagonal;
//Sets the fore color
shape.Fill.PatternFill.ForeColor = ColorObject.AliceBlue;
//Sets the back color
shape.Fill.PatternFill.BackColor = ColorObject.DarkSalmon;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```
///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide of the Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets the shape of the slide
IShape shape = slide.Shapes[0] as IShape;
//Sets the shape name.
shape.ShapeName = "Shape1";
//Retrieves the line format of the shape.
ILineFormat lineFormat = shape.LineFormat;
//Sets the dash style of the line format.
lineFormat.DashStyle = LineDashStyle.DashDotDot;
//Sets the weight of the line format.
lineFormat.Weight = 3;
//Sets the pattern fill type to shape
shape.Fill.FillType = FillType.Pattern;
//Chooses the type of pattern
shape.Fill.PatternFill.Pattern = PatternFillType.DashedDownwardDiagonal;
//Sets the fore color
shape.Fill.PatternFill.ForeColor = ColorObject.AliceBlue;
//Sets the back color
shape.Fill.PatternFill.BackColor = ColorObject.DarkSalmon;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
```

```
//The operation in Save under Xamarin varies between Windows Phone, Android
//and iOS platforms. Please refer presentation/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Removing the shapes

The shapes can be removed from a slide by its instance or by its index position in the shape collection. The following code example demonstrates how to remove the shapes from a slide.

C#

```
//Opens an existing Presentation from file system
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape.
IShape shape = slide.Shapes[0] as IShape;
//Removes the shape from the shape collection.
slide.Shapes.Remove(shape);
//Saves the Presentation to the file system.
pptxDoc.Save("Result.pptx");
//Closes the Presentation.
pptxDoc.Close();
```

VB.NET

```
'Opens an existing Presentation from file system
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Retrieves the first slide from Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Retrieves the first shape.
Dim shape As IShape = TryCast(slide.Shapes(0), IShape)
'Removes the shape from the shape collection.
slide.Shapes.Remove(shape)
'Saves the Presentation to the file system.
pptxDoc.Save("Result.pptx")
'Closes the Presentation.
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
```

```

//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape.
IShape shape = slide.Shapes[0] as IShape;
//Removes the shape from the shape collection.
slide.Shapes.Remove(shape);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Result";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape.
IShape shape = slide.Shapes[0] as IShape;
//Removes the shape from the shape collection.
slide.Shapes.Remove(shape);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation.
pptxDoc.Close();

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape.
IShape shape = slide.Shapes[0] as IShape;
//Removes the shape from the shape collection.
slide.Shapes.Remove(shape);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation

```

```
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Working with GroupShape

Creating a GroupShape

The shapes in a slide can be grouped into a single shape. The following code snippet demonstrates how to group different slide items into a single GroupShape.

C#

```
//Creates an instance for PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a group shape to the slide
IGroupShape groupShape = slide.GroupShapes.AddGroupShape(20, 20, 450, 300);
//Adds a TextBox to the group shape
groupShape.Shapes.AddTextBox(30, 25, 100, 100).TextBody.AddParagraph("My
TextBox");
//Gets the image stream
Stream pictureStream = File.Open("Image.png", FileMode.Open);
//Adds a picture to the group shape
groupShape.Shapes.AddPicture(pictureStream, 40, 100, 100, 100);
//Adds a shape to the group shape
groupShape.Shapes.AddShape(AutoShapeType.Rectangle, 200, 200, 90, 30);
//Save the presentation
pptxDoc.Save("Output.pptx");
//Close the presentation
pptxDoc.Close();
```

VB.NET

```
'Creates an instance for PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds a blank slide to presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds a group shape to the slide
Dim groupShape As IGroupShape = slide.GroupShapes.AddGroupShape(20, 20, 450,
300)
'Adds a TextBox to the group shape
groupShape.Shapes.AddTextBox(30, 25, 100, 100).TextBody.AddParagraph("My
TextBox")
'Gets the image stream
```

```

Dim pictureStream As Stream = File.Open("Image.png", FileMode.Open)
'Adds a picture to the group shape
groupShape.Shapes.AddPicture(pictureStream, 40, 100, 100, 100)
'Adds a shape to the group shape
groupShape.Shapes.AddShape(AutoShapeType.Rectangle, 200, 200, 90, 30)
'Saves the presentation
pptxDoc.Save("Output.pptx")
'Closes the presentation
pptxDoc.Close()

```

UWP

```

//Creates an instance for PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a group shape to the slide
IGroupShape groupShape = slide.GroupShapes.AddGroupShape(20, 20, 450, 300);
//Adds a TextBox to the group shape
groupShape.Shapes.AddTextBox(30, 25, 100, 100).TextBody.AddParagraph("My
TextBox");
//Gets the image stream
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream pictureStream =
assembly.GetManifestResourceStream("UWP.Data.Image.png");
//Adds a picture to the group shape
groupShape.Shapes.AddPicture(pictureStream, 40, 100, 100, 100);
//Adds a shape to the group shape
groupShape.Shapes.AddShape(AutoShapeType.Rectangle, 200, 200, 90, 30);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates an instance for PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a group shape to the slide
IGroupShape groupShape = slide.GroupShapes.AddGroupShape(20, 20, 450, 300);
//Adds a TextBox to the group shape
groupShape.Shapes.AddTextBox(30, 25, 100, 100).TextBody.AddParagraph("My
TextBox");
//Gets the image stream
FileStream pictureStream = new FileStream(imagePath, FileMode.Open);
//Adds a picture to the group shape
groupShape.Shapes.AddPicture(pictureStream, 40, 100, 100, 100);
//Adds a shape to the group shape

```

```
groupShape.Shapes.AddShape(AutoShapeType.Rectangle, 200, 200, 90, 30);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();
```

XAMARIN

```
//Creates an instance for PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a group shape to the slide
IGroupShape groupShape = slide.GroupShapes.AddGroupShape(20, 20, 450, 300);
//Adds a TextBox to the group shape
groupShape.Shapes.AddTextBox(30, 25, 100, 100).TextBody.AddParagraph("My
TextBox");
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets the image stream
Stream pictureStream = assembly.GetManifestResourceStream(picturePath);
//Adds a picture to the group shape
groupShape.Shapes.AddPicture(pictureStream, 40, 100, 100, 100);
//Adds a shape to the group shape
groupShape.Shapes.AddShape(AutoShapeType.Rectangle, 200, 200, 90, 30);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Iterating and modifying a particular GroupShape

You can iterate through the shape collection of a GroupShape. Below code snippet demonstrates how to iterate through the shapes of a GroupShape to modify it by removing a specific shape.

C#

```
//Opens a PowerPoint presentation with group shapes
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Retrieves the first slide
```

```

ISlide slide = pptxDoc.Slides[0];
//Retrieves the first group shape of the slide
IGroupShape groupShape = slide.GroupShapes[0];
//Creates an instance to hold shape collection
IShapes shapes = groupShape.Shapes;
//Iterates the shape collection to remove the picture in a group shape
foreach (IShape shape in shapes)
{
    if (shape.SlideItemType == SlideItemType.Picture)
    {
        shapes.Remove(shape);
        break;
    }
}
//Saves the presentation
pptxDoc.Save("Output.pptx");
//Closes the presentation
pptxDoc.Close();

```

VB.NET

```

'Opens a PowerPoint presentation with group shapes
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Retrieves the first slide
Dim slide As ISlide = pptxDoc.Slides(0)
'Retrieves the first group shape of the slide
Dim groupShape As IGroupShape = slide.GroupShapes(0)
'Creates an instance to hold shape collection
Dim shapes As IShapes = groupShape.Shapes
'Iterates shape collection to remove the picture in a group shape
For Each shape As IShape In shapes
    If shape.SlideItemType = SlideItemType.Picture Then
        shapes.Remove(shape)
    Exit For
    End If
Next
'Saves the presentation
pptxDoc.Save("Output.pptx")
'Closes the presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieves the first slide
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first group shape of the slide
IGroupShape groupShape = slide.GroupShapes[0];
//Creates an instance to hold shape collection

```

```

IShapes shapes = groupShape.Shapes;
//Iterates the shape collection to remove the picture in a group shape
foreach (IShape shape in shapes)
{
    if (shape.SlideItemType == SlideItemType.Picture)
    {
        shapes.Remove(shape);
        break;
    }
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first group shape of the slide
IGroupShape groupShape = slide.GroupShapes[0];
//Creates an instance to hold shape collection
IShapes shapes = groupShape.Shapes;
//Iterates the shape collection to remove the picture in a group shape
foreach (IShape shape in shapes)
{
    if (shape.SlideItemType == SlideItemType.Picture)
    {
        shapes.Remove(shape);
        break;
    }
}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the presentation
pptxDoc.Close();

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);

```



```

//Retrieves the first slide
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first group shape of the slide
IGroupShape groupShape = slide.GroupShapes[0];
//Creates an instance to hold shape collection
IShapes shapes = groupShape.Shapes;
//Iterates the shape collection to remove the picture in a group shape
foreach (IShape shape in shapes)
{
    if (shape.SlideItemType == SlideItemType.Picture)
    {
        shapes.Remove(shape);
        break;
    }
}
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);

```

Removing a GroupShape

GroupShape can be removed from a slide using its instance or by its index position in the GroupShape collection of the slide. Below code snippet explains how to remove a GroupShape from a slide.

C#

```

//Opens a PowerPoint presentation with group shapes
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Retrieves the first slide
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first group shape of the slide
IGroupShape groupShape = slide.GroupShapes[0];
//Removes the group shape from group shape collection
slide.GroupShapes.Remove(groupShape);
//Saves the presentation
pptxDoc.Save("Output.pptx");
//Closes the presentation
pptxDoc.Close();

```

VB.NET

```

'Opens a PowerPoint presentation with group shapes
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Retrieves the first slide
Dim slide As ISlide = pptxDoc.Slides(0)
'Retrieves the first group shape of the slide
Dim groupShape As IGroupShape = slide.GroupShapes(0)
'Removes the group shape from group shape collection
slide.GroupShapes.Remove(groupShape)
'Saves the presentation
pptxDoc.Save("Output.pptx")
'Closes the presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieves the first slide
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first group shape of the slide
IGroupShape groupShape = slide.GroupShapes[0];
//Removes the group shape from group shape collection
slide.GroupShapes.Remove(groupShape);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first group shape of the slide
IGroupShape groupShape = slide.GroupShapes[0];
//Removes the group shape from group shape collection
slide.GroupShapes.Remove(groupShape);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the presentation
pptxDoc.Close();

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
// Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
// Retrieves the first slide
ISlide slide = pptxDoc.Slides[0];
// Retrieves the first group shape of the slide
IGroupShape groupShape = slide.GroupShapes[0];
// Removes the group shape from group shape collection
slide.GroupShapes.Remove(groupShape);
// Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
// Save Presentation in stream format.
pptxDoc.Save(stream);
// Close the presentation
pptxDoc.Close();
stream.Position = 0;
// The operation in Save under Xamarin varies between Windows Phone, Android
// and iOS platforms. Please refer presentation/xamarin section for respective
// code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Note: The Presentation library do not have support for converting the auto-shapes with modified adjustment values in PowerPoint presentation to image or PDF conversion. To know more about adjustment values in PowerPoint shapes please click [here](#).

Working with images**Adding Images**

Essential Presentation library facilitates adding or modifying the images in a PowerPoint Presentation. The following code example demonstrates how to add a new image to the presentation.

C#

```

// Creates a instance of Presentation
IPresentation pptxDoc = Presentation.Create();
// Adds a blank slide.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
// Gets a picture as stream.
Stream pictureStream = File.Open("Image.png", FileMode.Open);
// Adds the picture to a slide by specifying its size and position.
IPicture picture = slide.Pictures.AddPicture(pictureStream, 0, 0, 250, 250);

```

```
//Saves the Presentation to the file system.
pptxDoc.Save("Sample.pptx");
//Dispose the image stream
pictureStream.Dispose();
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Creates a instance of Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds a blank slide.
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Gets a picture as stream.
Dim pictureStream As Stream = File.Open("Image.png", FileMode.Open)
'Adds the picture to a slide by specifying its size and position.
Dim picture As IPicture = slide.Pictures.AddPicture(pictureStream, 0, 0,
250, 250)
'Saves the Presentation to the file system.
pptxDoc.Save("Sample.pptx")
'Dispose the image stream
pictureStream.Dispose()
'Closes the Presentation
pptxDoc.Close()
```

UWP

```
//Creates a instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Gets a picture as stream.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream pictureStream =
assembly.GetManifestResourceStream("UWP.Data.Image.png");
//Adds the picture to a slide by specifying its size and position.
IPicture picture = slide.Pictures.AddPicture(pictureStream, 0, 0, 250, 250);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Creates a instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Gets a picture as stream.
```

```

FileStream pictureStream = new FileStream("Image.png", FileMode.Open);
//Adds the picture to a slide by specifying its size and position.
IPicture picture = slide.Pictures.AddPicture(pictureStream, 0, 0, 250, 250);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Dispose the image stream
pictureStream.Dispose();
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

//Creates a instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets a picture as stream.
Stream pictureStream = assembly.GetManifestResourceStream("Image.png");
//Adds the picture to a slide by specifying its size and position.
IPicture picture = slide.Pictures.AddPicture(pictureStream, 0, 0, 250, 250);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Replacing Images

The following code example demonstrates how to replace an existing image in a slide.

C#

```

//Opens an existing Presentation.
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Retrieves the first slide from the Presentation.
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first picture from the slide.
IPicture picture = slide.Pictures[0];
//Gets the new picture as stream.
Stream pictureStream = File.Open("Image.png", FileMode.Open);

```

```

//Creates instance for memory stream
MemoryStream memoryStream = new MemoryStream();
//Copies stream to memoryStream.
pictureStream.CopyTo(memoryStream);
//Replaces the existing image with new image.
picture.ImageData = memoryStream.ToArray();
//Saves the Presentation to the file system.
pptxDoc.Save("Output.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Opens an existing Presentation.
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Retrieves the first slide from the Presentation.
Dim slide As ISlide = pptxDoc.Slides(0)
'Retrieves the first picture from the slide.
Dim picture As IPicture = slide.Pictures(0)
'Gets the new picture as stream.
Dim pictureStream As Stream = File.Open("Image.png", FileMode.Open)
'Creates instance for memory stream
Dim memoryStream As New MemoryStream()
'Copies stream to memoryStream.
pictureStream.CopyTo(memoryStream)
'Replaces the existing image with new image.
picture.ImageData = memoryStream.ToArray()
'Saves the Presentation to the file system.
pptxDoc.Save("Output.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieves the first slide from the Presentation.
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first picture from the slide.
IPicture picture = slide.Pictures[0];
//Gets the new picture as stream.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets the new picture as stream.
Stream pictureStream =
assembly.GetManifestResourceStream("UWP.Data.Image.png");
//Creates instance for memory stream
MemoryStream memoryStream = new MemoryStream();
//Copies stream to memoryStream.
pictureStream.CopyTo(memoryStream);
//Replaces the existing image with new image.

```

```

picture.ImageData = memoryStream.ToArray();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from the Presentation.
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first picture from the slide.
IPicture picture = slide.Pictures[0];
//Gets the new picture as stream.
FileStream pictureStream = new FileStream("Image.png", FileMode.Open);
//Creates instance for memory stream
MemoryStream memoryStream = new MemoryStream();
//Copies stream to memoryStream.
pictureStream.CopyTo(memoryStream);
//Replaces the existing image with new image.
picture.ImageData = memoryStream.ToArray();
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from the Presentation.
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first picture from the slide.
IPicture picture = slide.Pictures[0];
//Gets the new picture as stream.
Stream pictureStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Image.png");
//Creates instance for memory stream
MemoryStream memoryStream = new MemoryStream();
//Copies stream to memoryStream.
pictureStream.CopyTo(memoryStream);

```

```

//Replaces the existing image with new image.
picture.ImageData = memoryStream.ToArray();
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Removing Images

The following code example demonstrates how to remove an existing image in a PowerPoint slide.

C#

```

//Opens an existing Presentation from file system.
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Iterates through the pictures collection and remove the picture named
"Image".
foreach (IPicture picture in slide.Pictures)
{
//Removes the picture from the slide.
slide.Pictures.Remove(picture);
break;
}
//Saves the Presentation to the file system.
pptxDoc.Save("Output.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Opens an existing Presentation from file system.
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Retrieves the first slide from Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Iterates through the pictures collection and removes the picture named
"Image".
For Each picture As IPicture In slide.Pictures
'Removes the picture from the slide.
slide.Pictures.Remove(picture)
Exit For

```


Next

```
'Saves the Presentation to the file system.
pptxDoc.Save("Output.pptx")
'Closes the Presentation
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Iterates through the pictures collection and remove the picture named
"Image".
foreach (IPicture picture in slide.Pictures)
{
//Removes the picture from the slide.
slide.Pictures.Remove(picture);
break;
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Iterates through the pictures collection and remove the picture named
"Image".
foreach (IPicture picture in slide.Pictures)
{
//Removes the picture from the slide.
slide.Pictures.Remove(picture);
break;
}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
```

```
pptxDoc.Close();
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Iterates through the pictures collection and remove the picture named
"Image".
foreach (IPicture picture in slide.Pictures)
{
//Removes the picture from the slide.
slide.Pictures.Remove(picture);
break;
}
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Working with PowerPoint Tables

Create a table by adding rows

Essential Presentation supports creating and editing tables in PowerPoint slides by adding rows. Refer to the following code example.

C#

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Initialize index values to add text to table cells
```

```

int rowIndex = 0, colIndex;
//Iterate row-wise cells and add text to it
foreach (IRow rows in table.Rows)
{
    colIndex = 0;
    foreach (ICell cell in rows.Cells)
    {
        cell.TextBody.AddParagraph("(" + rowIndex.ToString() + " , " +
        colIndex.ToString() + ")");
        colIndex++;
    }
    rowIndex++;
}
//Save the presentation
pptxDoc.Save("Sample.pptx");
//Close the presentation
pptxDoc.Close();

```

VB.NET

```

'Create a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Add slide to the presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a table to the slide
Dim table As ITable = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200)
'Initialize index values to add text to table cells
Dim rowIndex As Integer = 0, colIndex As Integer
'Iterate row-wise cells and add text to it
For Each rows As IRow In table.Rows
    colIndex = 0
    For Each cell As ICell In rows.Cells
        cell.TextBody.AddParagraph("(" + rowIndex.ToString() + " , " +
        colIndex.ToString() + ")")
        colIndex += 1
    Next
    rowIndex += 1
Next
'Save the presentation
pptxDoc.Save("Sample.pptx")
'Close the presentation
pptxDoc.Close()

```

UWP

```

//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Initialize index values to add text to table cells
int rowIndex = 0, colIndex;
//Iterate row-wise cells and add text to it
foreach (IRow rows in table.Rows)
{

```

```

colIndex = 0;
foreach (ICell cell in rows.Cells)
{
    cell.TextBody.AddParagraph("(" + rowIndex.ToString() + " , " +
    colIndex.ToString() + ")");
    colIndex++;
}
rowIndex++;
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Initialize index values to add text to table cells
int rowIndex = 0, colIndex;
//Iterate row-wise cells and add text to it
foreach (IRow rows in table.Rows)
{
    colIndex = 0;
    foreach (ICell cell in rows.Cells)
    {
        cell.TextBody.AddParagraph("(" + rowIndex.ToString() + " , " +
        colIndex.ToString() + ")");
        colIndex++;
    }
    rowIndex++;
}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();

```

XAMARIN

```

//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);

```

```

//Initialize index values to add text to table cells
int rowIndex = 0, colIndex;
//Iterate row-wise cells and add text to it
foreach (IRow rows in table.Rows)
{
    colIndex = 0;
    foreach (ICell cell in rows.Cells)
    {
        cell.TextBody.AddParagraph("(" + rowIndex.ToString() + " , " +
            colIndex.ToString() + ")");
        colIndex++;
    }
    rowIndex++;
}
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);

```

Create a table by adding columns

The following code example demonstrates how to create a simple table in a PowerPoint slide by adding columns.

C#

```

//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Initialize index values to add text to table cells
int row = 0, col;
//Iterate row-wise cells and add text to it
foreach (IColumn columns in table.Columns)
{
    col = 0;
    foreach (ICell cell in columns.Cells)
    {
        cell.TextBody.AddParagraph("(" + row.ToString() + " , " + col.ToString() +
            ")");
    }
}

```

```
col++;
}
row++;
}
//Save the presentation
pptxDoc.Save("Sample.pptx");
//Close the presentation
pptxDoc.Close();
```

VB.NET

```
'Create a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Add slide to the presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a table to the slide
Dim table As ITable = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200)
'Initialize index values to add text to the table cells
Dim row As Integer = 0, col As Integer
'Iterate row-wise cells and add text to it
For Each columns As IColumn In table.Columns
col = 0
For Each cell As ICell In columns.Cells
cell.TextBody.AddParagraph("(" + row.ToString() + " , " + col.ToString() +
")")
col += 1
Next
row += 1
Next
'Save the presentation
pptxDoc.Save("Sample.pptx")
'Close the presentation
pptxDoc.Close()
```

UWP

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Initialize index values to add text to table cells
int row = 0, col;
//Iterate row-wise cells and add text to it
foreach (IColumn columns in table.Columns)
{
col = 0;
foreach (ICell cell in columns.Cells)
{
cell.TextBody.AddParagraph("(" + row.ToString() + " , " + col.ToString() +
")");
col++;
}
row++;
}
```

```
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Initialize index values to add text to table cells
int row = 0, col;
//Iterate row-wise cells and add text to it
foreach (IColumn columns in table.Columns)
{
    col = 0;
    foreach (ICell cell in columns.Cells)
    {
        cell.TextBody.AddParagraph("(" + row.ToString() + " , " + col.ToString() +
            ")");
        col++;
    }
    row++;
}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();
```

XAMARIN

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Initialize index values to add text to table cells
int row = 0, col;
//Iterate row-wise cells and add text to it
foreach (IColumn columns in table.Columns)
{
    col = 0;
    foreach (ICell cell in columns.Cells)
    {
```

```

cell.TextBody.AddParagraph("(" + row.ToString() + " , " + col.ToString() +
")");
col++;
}
row++;
}
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Append a new row at the end of table

You can append new rows at the end of an existing PowerPoint table. Refer to the following code sample.

C#

```

//Open an existing PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Table.pptx");
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Add or append a new row at the end of table
IRow row = table.Rows.Add();
//Iterate row-wise cells and add text to it
foreach (ICell cell in row.Cells)
{
cell.TextBody.AddParagraph(table.Rows.IndexOf(row).ToString());
}
//Save the presentation
pptxDoc.Save("Sample.pptx");
//Close the presentation
pptxDoc.Close();

```

VB.NET

```

'Open an existing PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Table.pptx")
'Get a table in the slide
Dim table As ITable = TryCast(pptxDoc.Slides(0).Shapes(0), ITable)
'Add a new row at the end of table

```



```

Dim row As IRow = table.Rows.Add()
'Iterate row-wise cells and add text to it
For Each cell As ICell In row.Cells
cell.TextBody.AddParagraph(table.Rows.IndexOf(row).ToString())
Next
'Save the presentation
pptxDoc.Save("Sample.pptx")
'Close the presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Add or append a new row at the end of table
IRow row = table.Rows.Add();
//Iterate row-wise cells and add text to it
foreach (ICell cell in row.Cells)
{
cell.TextBody.AddParagraph(table.Rows.IndexOf(row).ToString());
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Add or append a new row at the end of table
IRow row = table.Rows.Add();
//Iterate row-wise cells and add text to it
foreach (ICell cell in row.Cells)
{
cell.TextBody.AddParagraph(table.Rows.IndexOf(row).ToString());
}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Table.pptx", FileMode.Create);

```

```
pptxDoc.Save(outputStream);
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Add or append a new row at the end of table
IRow row = table.Rows.Add();
//Iterate row-wise cells and add text to it
foreach (ICell cell in row.Cells)
{
    cell.TextBody.AddParagraph(table.Rows.IndexOf(row).ToString());
}
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Copy an existing row to the end of table

You can copy an existing row to the end of a table. Refer to the following code example.

C#

```
//Open an existing PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Table.pptx");
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Copy the first row to the end of table
table.Rows.Add(table.Rows[0].Clone());
//Save the presentation
pptxDoc.Save("Sample.pptx");
//Close the presentation
pptxDoc.Close();
```

VB.NET

```

'Open an existing PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Table.pptx")
'Get the table in the slide
Dim table As ITable = TryCast(pptxDoc.Slides(0).Shapes(0), ITable)
'Copy the first row to the end of table
table.Rows.Add(table.Rows(0).Clone())
'Save the presentation
pptxDoc.Save("Sample.pptx")
'Close the presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Copy the first row to the end of table
table.Rows.Add(table.Rows[0].Clone());
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Copy the first row to the end of table
table.Rows.Add(table.Rows[0].Clone());
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Table.pptx", FileMode.Create);
pptxDoc.Save(outputStream);

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;

```

```

Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Copy the first row to the end of table
table.Rows.Add(table.Rows[0].Clone());
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Insert a row in table

You can insert a row at the specified index position of a table. Refer to the following code example.

C#

```

//Open an existing PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Table.pptx");
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Insert a row at the specified index. Here, the existing first row at index
0 is copied and inserted at row index 1
table.Rows.Insert(1, table.Rows[0].Clone());
//Save the presentation
pptxDoc.Save("Sample.pptx");
//Close the presentation
pptxDoc.Close();

```

VB.NET

```

'Open an existing PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Table.pptx")
'Get a table in the slide
Dim table As ITable = TryCast(pptxDoc.Slides(0).Shapes(0), ITable)
'Insert a row at the specified index. Here, the existing first row at index
0 is copied and inserted at row index 1.
table.Rows.Insert(1, table.Rows(0).Clone())
'Save the presentation
pptxDoc.Save("Sample.pptx")
'Close the presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Insert a row at the specified index. Here, the existing first row at index 0 is copied and inserted at row index 1
table.Rows.Insert(1, table.Rows[0].Clone());
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Insert a row at the specified index. Here, the existing first row at index 0 is copied and inserted at row index 1
table.Rows.Insert(1, table.Rows[0].Clone());
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Table.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;

```

```

//Insert a row at the specified index. Here, the existing first row at index 0 is copied and inserted at row index 1
table.Rows.Insert(1, table.Rows[0].Clone());
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Table.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Table.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);

```

Append a new column at the end table

You can append new column to a table. Refer to the following code example.

C#

```

//Open an existing PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Table.pptx");
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Add or append a new column at the end of table
IColumn column = table.Columns.Add();
//Iterate row-wise cells and add text to it
foreach (ICell cell in column.Cells)
{
    cell.TextBody.AddParagraph(table.Columns.IndexOf(column).ToString());
}
//Save the presentation
pptxDoc.Save("Sample.pptx");
//Close the presentation
pptxDoc.Close();

```

VB.NET

```

'Open an existing PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Table.pptx")
'Get a table in the slide
Dim table As ITable = TryCast(pptxDoc.Slides(0).Shapes(0), ITable)
'Add or append a new column at the end of table
Dim column As IColumn = table.Columns.Add()
'Iterate row-wise cells and add text to it
For Each cell As ICell In column.Cells
    cell.TextBody.AddParagraph(table.Columns.IndexOf(column).ToString())
Next
'Save the presentation
pptxDoc.Save("Sample.pptx")
'Close the presentation

```

```
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Add or append a new column at the end of table
IColumn column = table.Columns.Add();
//Iterate row-wise cells and add text to it
foreach (ICell cell in column.Cells)
{
    cell.TextBody.AddParagraph(table.Columns.IndexOf(column).ToString());
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Add or append a new column at the end of table
IColumn column = table.Columns.Add();
//Iterate row-wise cells and add text to it
foreach (ICell cell in column.Cells)
{
    cell.TextBody.AddParagraph(table.Columns.IndexOf(column).ToString());
}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Table.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();
```

XAMARIN

```
/"App" is the class of Portable project.
```

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Add or append a new column at the end of table
IColumn column = table.Columns.Add();
//Iterate row-wise cells and add text to it
foreach (ICell cell in column.Cells)
{
cell.TextBody.AddParagraph(table.Columns.IndexOf(column).ToString());
}
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Copy an existing column to the end of table

You can copy an existing column and append it to the end of table. Refer to the following code example.

C#

```

//Open an existing PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Table.pptx");
//Get a table from the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Copy the column and append it to the end of table
table.Columns.Add(table.Columns[0].Clone());
//Save the presentation
pptxDoc.Save("Sample.pptx");
//Close the presentation
pptxDoc.Close();

```

VB.NET

```

'Open an existing PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Table.pptx")
'Get a table from the slide
Dim table As ITable = TryCast(pptxDoc.Slides(0).Shapes(0), ITable)
'Copy the column and append it to the end of table
table.Columns.Add(table.Columns(0).Clone())

```



```
'Save the presentation
pptxDoc.Save("Sample.pptx")
'Close the presentation
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Get a table from the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Copy the column and append it to the end of table
table.Columns.Add(table.Columns[0].Clone());
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Copy the column and append it to the end of table
table.Columns.Add(table.Columns[0].Clone());
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Table.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();
```

XAMARIN

```
/"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get a table in the slide
```

```

ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Copy the column and append it to the end of table
table.Columns.Add(table.Columns[0].Clone());
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Insert a column to a table

You can insert a column at the specified index position of a table. Refer to the following code example.

C#

```

//Open an existing PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Table.pptx");
//Get the table from the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Insert a column at the specified index. Here, the existing first column at
index 0 is copied and inserted at column index 1
table.Columns.Insert(1, table.Columns[0].Clone());
//Save the presentation
pptxDoc.Save("Sample.pptx");
//Close the presentation
pptxDoc.Close();

```

VB.NET

```

'Open an existing PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Table.pptx")
'Get the table from the slide
Dim table As ITable = TryCast(pptxDoc.Slides(0).Shapes(0), ITable)
'Insert a column at the specified index. Here, the existing first column at
index 0 is copied and inserted at column index 1.
table.Columns.Insert(1, table.Columns(0).Clone())
'Save the presentation
pptxDoc.Save("Sample.pptx")
'Close the presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();

```

```

openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Get the table from the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Insert a column at the specified index. Here, the existing first column at index 0 is copied and inserted at column index 1
table.Columns.Insert(1, table.Columns[0].Clone());
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() { ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Insert a column at the specified index. Here, the existing first column at index 0 is copied and inserted at column index 1
table.Columns.Insert(1, table.Columns[0].Clone());
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Table.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Insert a column at the specified index. Here, the existing first column at index 0 is copied and inserted at column index 1
table.Columns.Insert(1, table.Columns[0].Clone());
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);

```

```
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Get the actual height of the table

The table height expands with the content added to it. The Essential Presentation library allows you to get this actual height or rendered height of the table. This property is a calculated value based on the content added to the table cells.

The following code example demonstrates how to get the actual height of a PowerPoint table.

C#

```
//Opens existing PowerPoint file
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Opens slide in the presentation
ISlide slide = pptxDoc.Slides[0];
//Open Table in the slide to make changes
ITable table = slide.Shapes[0] as ITable;
//Changing the paragraph content in the table
table.Rows[0].Cells[0].TextBody.AddParagraph("Hello World");
//Get the dynamic height of the table
float height=table.GetActualHeight();
//Save the presentation
pptxDoc.Save("Table.pptx");
//Close the presentation
pptxDoc.Close();
```

VB.NET

```
'Open an existing PowerPoint file
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Get the table from the slide
Dim table As ITable = TryCast(pptxDoc.Slides(0).Shapes(0), ITable)
'Changing the paragraph content in the table
table.Rows[0].Cells[0].TextBody.AddParagraph("Hello World");
'Get the dynamic height of table
Dim height As float = table.GetActualHeight()
'Save the presentation
pptxDoc.Save("Table.pptx")
'Close the presentation
pptxDoc.Close()
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
```

```

FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Initialize Presentation renderer
pptxDoc.PresentationRenderer = new PresentationRenderer();
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Changing the paragraph content in the table
table.Rows[0].Cells[0].TextBody.AddParagraph("Hello World");
//Get the dynamic height of the table
float height=table.GetActualHeight();
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Table.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Initialize Presentation renderer
pptxDoc.PresentationRenderer = new PresentationRenderer();
//Get a table in the slide
ITable table = pptxDoc.Slides[0].Shapes[0] as ITable;
//Changing the paragraph content in the table
table.Rows[0].Cells[0].TextBody.AddParagraph("Hello World");
//Get the dynamic height of the table
float height=table.GetActualHeight();
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Note: Getting the actual height of the table is not supported in UWP platform.

Applying table formatting

You can format a table to change its appearance by customizing the table border, cell background, cell margins etc. The following code example demonstrates how to apply the custom table formatting.

C#

```

//Creates instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Retrieves each cell and fills text content to the cell.
ICell cell = table[0, 0];
//Sets the column width for a cell; this sets the width for entire column
cell.ColumnWidth = 400;
//Sets the margin for the cell.
cell.TextBody.MarginBottom = 0;
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color.SystemColor = Color.Orange;
cell.TextBody.AddParagraph("First Row and First Column");
cell = table[0, 1];
//Sets the margin for the cell.
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color.SystemColor = Color.BlueViolet;
cell.TextBody.AddParagraph("First Row and Second Column");
cell = table[1, 0];
//Sets the margin for the cell.
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color.SystemColor = Color.SandyBrown;
cell.TextBody.AddParagraph("Second Row and First Column");
cell = table[1, 1];
//Sets the margin for the cell.
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color.SystemColor = Color.Silver;
cell.TextBody.AddParagraph("Second Row and Second Column");
//Saves the Presentation
pptxDoc.Save("Table.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Creates instance of PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds slide to the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds table to the slide

```

```

Dim table As ITable = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200)
'Retrieves each cell and fills text content to the cell.
Dim cell As ICell = table(0, 0)
'Sets the column width for a cell; this sets the width for entire column
cell.ColumnWidth = 400
'Sets the margin for the cell.
cell.TextBody.MarginBottom = 0
cell.TextBody.MarginLeft = 58
cell.TextBody.MarginRight = 29
cell.TextBody.MarginTop = 65
'Sets the back color for the cell.
cell.Fill.SolidFill.Color.SystemColor = Color.Orange
cell.TextBody.AddParagraph("First Row and First Column")
cell = table(0, 1)
'Sets the margin for the cell.
cell.TextBody.MarginLeft = 58
cell.TextBody.MarginRight = 29
cell.TextBody.MarginTop = 65
'Sets the back color for the cell.
cell.Fill.SolidFill.Color.SystemColor = Color.BlueViolet
cell.TextBody.AddParagraph("First Row and Second Column")
cell = table(1, 0)
'Sets the margin for the cell.
cell.TextBody.MarginLeft = 58
cell.TextBody.MarginRight = 29
cell.TextBody.MarginTop = 65
'Sets the back color for the cell.
cell.Fill.SolidFill.Color.SystemColor = Color.SandyBrown
cell.TextBody.AddParagraph("Second Row and First Column")
cell = table(1, 1)
'Sets the margin for the cell.
cell.TextBody.MarginLeft = 58
cell.TextBody.MarginRight = 29
cell.TextBody.MarginTop = 65
'Sets the back color for the cell.
cell.Fill.SolidFill.Color.SystemColor = Color.Silver
cell.TextBody.AddParagraph("Second Row and Second Column")
'Saves the Presentation
pptxDoc.Save("Table.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Creates instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Retrieves each cell and fills text content to the cell.
ICell cell = table[0, 0];
//Sets the column width for a cell; this sets the width for entire column
cell.ColumnWidth = 400;
//Sets the margin for the cell.
cell.TextBody.MarginBottom = 0;

```

```

cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color = ColorObject.Orange;
cell.TextBody.AddParagraph("First Row and First Column");
cell = table[0, 1];
//Sets the margin for the cell.
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color = ColorObject.BlueViolet;
cell.TextBody.AddParagraph("First Row and Second Column");
cell = table[1, 0];
//Sets the margin for the cell.
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color = ColorObject.SandyBrown;
cell.TextBody.AddParagraph("Second Row and First Column");
cell = table[1, 1];
//Sets the margin for the cell.
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color = ColorObject.Silver;
cell.TextBody.AddParagraph("Second Row and Second Column");
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Table";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Retrieves each cell and fills text content to the cell.
ICell cell = table[0, 0];
//Sets the column width for a cell; this sets the width for entire column
cell.ColumnWidth = 400;
//Sets the margin for the cell.
cell.TextBody.MarginBottom = 0;
cell.TextBody.MarginLeft = 58;

```



```

cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color = ColorObject.Orange;
cell.TextBody.AddParagraph("First Row and First Column");
cell = table[0, 1];
//Sets the margin for the cell.
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color = ColorObject.BlueViolet;
cell.TextBody.AddParagraph("First Row and Second Column");
cell = table[1, 0];
//Sets the margin for the cell.
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color = ColorObject.SandyBrown;
cell.TextBody.AddParagraph("Second Row and First Column");
cell = table[1, 1];
//Sets the margin for the cell.
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color = ColorObject.Silver;
cell.TextBody.AddParagraph("Second Row and Second Column");
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Table.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

//Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Retrieves each cell and fills text content to the cell.
ICell cell = table[0, 0];
//Sets the column width for a cell; this sets the width for entire column
cell.ColumnWidth = 400;
//Sets the margin for the cell.
cell.TextBody.MarginBottom = 0;
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color = ColorObject.Orange;
cell.TextBody.AddParagraph("First Row and First Column");
cell = table[0, 1];

```

```

//Sets the margin for the cell.
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color = ColorObject.BlueViolet;
cell.TextBody.AddParagraph("First Row and Second Column");
cell = table[1, 0];
//Sets the margin for the cell.
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color= ColorObject.SandyBrown;
cell.TextBody.AddParagraph("Second Row and First Column");
cell = table[1, 1];
//Sets the margin for the cell.
cell.TextBody.MarginLeft = 58;
cell.TextBody.MarginRight = 29;
cell.TextBody.MarginTop = 65;
//Sets the back color for the cell.
cell.Fill.SolidFill.Color = ColorObject.Silver;
cell.TextBody.AddParagraph("Second Row and Second Column");
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Applying table styles

You can format a table by applying pre-defined table styles. The following code example demonstrates how to apply predefined styles to a table.

C#

```

//Creates instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds table to the slide
ITable table = slide.Shapes.AddTable(3, 3, 100, 120, 300, 200);
table.BuiltInStyle = BuiltInTableStyle.ThemedStyle2Accent4;
table.HasBandedRows = false;
table.HasHeaderRow = false;
table.HasBandedColumns = true;

```

```

table.HasFirstColumn = true;
table.HasLastColumn = true;
table.HasTotalRow = true;
//Retrieves each cell and fills text content to the cell.
ICell cell = table[0, 0];
cell.TextBody.AddParagraph("First Row and First Column");
cell = table[0, 1];
cell.TextBody.AddParagraph("First Row and Second Column");
cell = table[0, 2];
cell.TextBody.AddParagraph("First Row and Third Column");
cell = table[1, 0];
cell.TextBody.AddParagraph("Second Row and First Column");
cell = table[1, 1];
cell.TextBody.AddParagraph("Second Row and Second Column");
cell = table[1, 2];
cell.TextBody.AddParagraph("Second Row and Third Column");
cell = table[2, 0];
cell.TextBody.AddParagraph("Third Row and First Column");
cell = table[2, 1];
cell.TextBody.AddParagraph("Third Row and Second Column");
cell = table[2, 2];
cell.TextBody.AddParagraph("Third Row and Third Column");
//Adds description to table shape
table.Description = "Table arrangement";
//Saves the Presentation
pptxDoc.Save("Table.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Creates instance of PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds slide to the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds table to the slide
Dim table As ITable = slide.Shapes.AddTable(3, 3, 100, 120, 300, 200)
table.BuiltInStyle = BuiltInTableStyle.ThemedStyle2Accent4
table.HasBandedRows = False
table.HasHeaderRow = False
table.HasBandedColumns = True
table.HasFirstColumn = True
table.HasLastColumn = True
table.HasTotalRow = True
'Retrieves each cell and fills text content to the cell.
Dim cell As ICell = table(0, 0)
cell.TextBody.AddParagraph("First Row and First Column")
cell = table(0, 1)
cell.TextBody.AddParagraph("First Row and Second Column")
cell = table(0, 2)
cell.TextBody.AddParagraph("First Row and Third Column")
cell = table(1, 0)
cell.TextBody.AddParagraph("Second Row and First Column")
cell = table(1, 1)
cell.TextBody.AddParagraph("Second Row and Second Column")
cell = table(1, 2)

```

```

cell.TextBody.AddParagraph("Second Row and Third Column")
cell = table(2, 0)
cell.TextBody.AddParagraph("Third Row and First Column")
cell = table(2, 1)
cell.TextBody.AddParagraph("Third Row and Second Column")
cell = table(2, 2)
cell.TextBody.AddParagraph("Third Row and Third Column")
'Adds description to table shape
table.Description = "Table arrangement"
'Saves the Presentation
pptxDoc.Save("Table.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Creates instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds table to the slide
ITable table = slide.Shapes.AddTable(3, 3, 100, 120, 300, 200);
table.BuiltInStyle = BuiltInTableStyle.ThemedStyle2Accent4;
table.HasBandedRows = false;
table.HasHeaderRow = false;
table.HasBandedColumns = true;
table.HasFirstColumn = true;
table.HasLastColumn = true;
table.HasTotalRow = true;
//Retrieves each cell and fills text content to the cell.
ICell cell = table[0, 0];
cell.TextBody.AddParagraph("First Row and First Column");
cell = table[0, 1];
cell.TextBody.AddParagraph("First Row and Second Column");
cell = table[0, 2];
cell.TextBody.AddParagraph("First Row and Third Column");
cell = table[1, 0];
cell.TextBody.AddParagraph("Second Row and First Column");
cell = table[1, 1];
cell.TextBody.AddParagraph("Second Row and Second Column");
cell = table[1, 2];
cell.TextBody.AddParagraph("Second Row and Third Column");
cell = table[2, 0];
cell.TextBody.AddParagraph("Third Row and First Column");
cell = table[2, 1];
cell.TextBody.AddParagraph("Third Row and Second Column");
cell = table[2, 2];
cell.TextBody.AddParagraph("Third Row and Third Column");
//Adds description to table shape
table.Description = "Table arrangement";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Table";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });

```

```
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Creates instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds table to the slide
ITable table = slide.Shapes.AddTable(3, 3, 100, 120, 300, 200);
table.BuiltInStyle = BuiltInTableStyle.ThemedStyle2Accent4;
table.HasBandedRows = false;
table.HasHeaderRow = false;
table.HasBandedColumns = true;
table.HasFirstColumn = true;
table.HasLastColumn = true;
table.HasTotalRow = true;
//Retrieves each cell and fills text content to the cell.
ICell cell = table[0, 0];
cell.TextBody.AddParagraph("First Row and First Column");
cell = table[0, 1];
cell.TextBody.AddParagraph("First Row and Second Column");
cell = table[0, 2];
cell.TextBody.AddParagraph("First Row and Third Column");
cell = table[1, 0];
cell.TextBody.AddParagraph("Second Row and First Column");
cell = table[1, 1];
cell.TextBody.AddParagraph("Second Row and Second Column");
cell = table[1, 2];
cell.TextBody.AddParagraph("Second Row and Third Column");
cell = table[2, 0];
cell.TextBody.AddParagraph("Third Row and First Column");
cell = table[2, 1];
cell.TextBody.AddParagraph("Third Row and Second Column");
cell = table[2, 2];
cell.TextBody.AddParagraph("Third Row and Third Column");
//Adds description to table shape
table.Description = "Table arrangement";
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Table.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```
//Creates instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds table to the slide
ITable table = slide.Shapes.AddTable(3, 3, 100, 120, 300, 200);
table.BuiltInStyle = BuiltInTableStyle.ThemedStyle2Accent4;
```

```

table.HasBandedRows = false;
table.HasHeaderRow = false;
table.HasBandedColumns = true;
table.HasFirstColumn = true;
table.HasLastColumn = true;
table.HasTotalRow = true;
//Retrieves each cell and fills text content to the cell.
ICell cell = table[0, 0];
cell.TextBody.AddParagraph("First Row and First Column");
cell = table[0, 1];
cell.TextBody.AddParagraph("First Row and Second Column");
cell = table[0, 2];
cell.TextBody.AddParagraph("First Row and Third Column");
cell = table[1, 0];
cell.TextBody.AddParagraph("Second Row and First Column");
cell = table[1, 1];
cell.TextBody.AddParagraph("Second Row and Second Column");
cell = table[1, 2];
cell.TextBody.AddParagraph("Second Row and Third Column");
cell = table[2, 0];
cell.TextBody.AddParagraph("Third Row and First Column");
cell = table[2, 1];
cell.TextBody.AddParagraph("Third Row and Second Column");
cell = table[2, 2];
cell.TextBody.AddParagraph("Third Row and Third Column");
//Adds description to table shape
table.Description = "Table arrangement";
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Modifying the table

The following code example demonstrates how to modify the table in existing PowerPoint Presentation

C#

```

//Creates instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Table.pptx");
//Gets slide from the Presentation
ISlide slide = pptxDoc.Slides[0];

```

```

//Gets table from slide
ITable table = slide.Shapes[0] as ITable;
//Modifies the table width
table.Width = 450;
//Changes the built in style of the table
table.BuiltInStyle = BuiltInTableStyle.DarkStyle1Accent2;
//Sets text content to the cell
table.Rows[0].Cells[0].TextBody.AddParagraph("Row1 Cell1");
//Saves the Presentation
pptxDoc.Save("TableModified.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Creates instance of PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Table.pptx")
'Gets slide from the Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Gets table from slide
Dim table As ITable = TryCast(slide.Shapes(0), ITable)
'Modifies the table width
table.Width = 450
'Changes the built in style of the table
table.BuiltInStyle = BuiltInTableStyle.DarkStyle1Accent2
'Sets text content to the cell
table.Rows(0).Cells(0).TextBody.AddParagraph("Row1 Cell1")
'Saves the Presentation
pptxDoc.Save("TableModified.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Gets slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets table from slide
ITable table = slide.Shapes[0] as ITable;
//Modifies the table width
table.Width = 450;
//Changes the built in style of the table
table.BuiltInStyle = BuiltInTableStyle.DarkStyle1Accent2;
//Sets text content to the cell
table.Rows[0].Cells[0].TextBody.AddParagraph("Row1 Cell1");
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "TableModified";

```

```

savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets table from slide
ITable table = slide.Shapes[0] as ITable;
//Modifies the table width
table.Width = 450;
//Changes the built in style of the table
table.BuiltInStyle = BuiltInTableStyle.DarkStyle1Accent2;
//Sets text content to the cell
table.Rows[0].Cells[0].TextBody.AddParagraph("Row1 Cell1");
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("TableModified.pptx",
FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();

```

XAMARIN

```

/"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets table from slide
ITable table = slide.Shapes[0] as ITable;
//Modifies the table width
table.Width = 450;
//Changes the built in style of the table
table.BuiltInStyle = BuiltInTableStyle.DarkStyle1Accent2;
//Sets text content to the cell
table.Rows[0].Cells[0].TextBody.AddParagraph("Row1 Cell1");
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;

```



```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Merging the cells

The following code example shows how to merge cells in a table.

C#

```

//Creates instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Retrieves first cell.
ICell cell = table[0, 0];
//Sets the column span value to merge the cell.
cell.ColumnSpan = 2;
//Retrieves each cell and fills text content to the cell.
cell.TextBody.AddParagraph("First Row and First Column");
cell = table[0, 1];
cell.TextBody.AddParagraph("First Row and Second Column");
cell = table[1, 0];
cell.TextBody.AddParagraph("Second Row and First Column");
cell = table[1, 1];
cell.TextBody.AddParagraph("Second Row and Second Column");
//Gives simple description to table shape
table.Description = "Table arrangement";
//Saves the Presentation
pptxDoc.Save("Table.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Creates instance of PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds slide to the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds table to the slide
Dim table As ITable = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200)
'Retrieves first cell.
Dim cell As ICell = table(0, 0)
'Sets the column span value to merge the cell.
cell.ColumnSpan = 2
'Retrieves each cell and fills text content to the cell.
cell.TextBody.AddParagraph("First Row and First Column")
cell = table(0, 1)
cell.TextBody.AddParagraph("First Row and Second Column")

```

```

cell = table(1, 0)
cell.TextBody.AddParagraph("Second Row and First Column")
cell = table(1, 1)
cell.TextBody.AddParagraph("Second Row and Second Column")
'Gives simple description to table shape
table.Description = "Table arrangement"
'Saves the Presentation
pptxDoc.Save("Table.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Creates instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Retrieves first cell.
ICell cell = table[0, 0];
//Sets the column span value to merge the cell.
cell.ColumnSpan = 2;
//Retrieves each cell and fills text content to the cell.
cell.TextBody.AddParagraph("First Row and First Column");
cell = table[0, 1];
cell.TextBody.AddParagraph("First Row and Second Column");
cell = table[1, 0];
cell.TextBody.AddParagraph("Second Row and First Column");
cell = table[1, 1];
cell.TextBody.AddParagraph("Second Row and Second Column");
//Gives simple description to table shape
table.Description = "Table arrangement";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Table";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Retrieves first cell.
ICell cell = table[0, 0];
//Sets the column span value to merge the cell.
cell.ColumnSpan = 2;

```

```
//Retrieves each cell and fills text content to the cell.
cell.TextBody.AddParagraph("First Row and First Column");
cell = table[0, 1];
cell.TextBody.AddParagraph("First Row and Second Column");
cell = table[1, 0];
cell.TextBody.AddParagraph("Second Row and First Column");
cell = table[1, 1];
cell.TextBody.AddParagraph("Second Row and Second Column");
//Gives simple description to table shape
table.Description = "Table arrangement";
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```
//Creates instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds table to the slide
ITable table = slide.Shapes.AddTable(2, 2, 100, 120, 300, 200);
//Retrieves first cell.
ICell cell = table[0, 0];
//Sets the column span value to merge the cell.
cell.ColumnSpan = 2;
//Retrieves each cell and fills text content to the cell.
cell.TextBody.AddParagraph("First Row and First Column");
cell = table[0, 1];
cell.TextBody.AddParagraph("First Row and Second Column");
cell = table[1, 0];
cell.TextBody.AddParagraph("Second Row and First Column");
cell = table[1, 1];
cell.TextBody.AddParagraph("Second Row and Second Column");
//Gives simple description to table shape
table.Description = "Table arrangement";
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",  
"application/vnd.openxmlformats-officedocument.presentationml.presentation",  
stream);
```

Removing the table

You can remove a table from a slide by its instance or by its index position in the shape collection. The following code example demonstrates removing a table in a slide.

C#

```
//Creates instance of PowerPoint Presentation  
IPresentation pptxDoc = Presentation.Open("Table.pptx");  
//Gets slide from the Presentation  
ISlide slide = pptxDoc.Slides[0];  
//Gets the table from slide  
ITable table = slide.Shapes[0] as ITable;  
//Removes table from shape collection  
slide.Shapes.Remove(table);  
//Saves the Presentation  
pptxDoc.Save("TableRemoved.pptx");  
//Closes the Presentation  
pptxDoc.Close();
```

VB.NET

```
'Creates instance of PowerPoint Presentation  
Dim pptxDoc As IPresentation = Presentation.Open("Table.pptx")  
'Gets slide from the Presentation  
Dim slide As ISlide = pptxDoc.Slides(0)  
'Gets the table from slide  
Dim table As ITable = TryCast(slide.Shapes(0), ITable)  
'Removes table from shape collection  
slide.Shapes.Remove(table)  
'Saves the Presentation  
pptxDoc.Save("TableRemoved.pptx")  
'Closes the Presentation  
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker  
FileOpenPicker openPicker = new FileOpenPicker();  
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;  
openPicker.FileTypeFilter.Add(".pptx");  
//Creates a storage file from FileOpenPicker  
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();  
//Loads or open an PowerPoint Presentation  
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);  
//Gets slide from the Presentation  
ISlide slide = pptxDoc.Slides[0];  
//Gets the table from slide  
ITable table = slide.Shapes[0] as ITable;  
//Removes table from shape collection  
slide.Shapes.Remove(table);  
//Initializes FileSavePicker
```

```

FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "TableRemoved";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets the table from slide
ITable table = slide.Shapes[0] as ITable;
//Removes table from shape collection
slide.Shapes.Remove(table);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("TableModified.pptx",
FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets the table from slide
ITable table = slide.Shapes[0] as ITable;
//Removes table from shape collection
slide.Shapes.Remove(table);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else

```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("Table.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Working with Charts

Creating a Chart from scratch

An instance of **IOfficeChart** can be used to create or modify the charts in PowerPoint Presentation. The following code example demonstrates how to create a simple chart by adding data from scratch.

C#

```
//Creates a Presentation instance
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds chart to the slide with position and size
IPresentationChart chart = slide.Charts.AddChart(100, 10, 700, 500);
//Specifies the chart title
chart.ChartTitle = "Sales Analysis";
//Sets chart data - Row1
chart.ChartData.SetValue(1, 2, "Jan");
chart.ChartData.SetValue(1, 3, "Feb");
chart.ChartData.SetValue(1, 4, "March");
//Sets chart data - Row2
chart.ChartData.SetValue(2, 1, 2010);
chart.ChartData.SetValue(2, 2, 60);
chart.ChartData.SetValue(2, 3, 70);
chart.ChartData.SetValue(2, 4, 80);
//Sets chart data - Row3
chart.ChartData.SetValue(3, 1, 2011);
chart.ChartData.SetValue(3, 2, 80);
chart.ChartData.SetValue(3, 3, 70);
chart.ChartData.SetValue(3, 4, 60);
//Sets chart data - Row4
chart.ChartData.SetValue(4, 1, 2012);
chart.ChartData.SetValue(4, 2, 60);
chart.ChartData.SetValue(4, 3, 70);
chart.ChartData.SetValue(4, 4, 80);
//Creates a new chart series with the name
IOfficeChartSerie seriesJan = chart.Series.Add("Jan");
//Sets the data range of chart series - start row, start column, end row,
end column
seriesJan.Values = chart.ChartData[2, 2, 4, 2];
//Creates a new chart series with the name
IOfficeChartSerie seriesFeb = chart.Series.Add("Feb");
//Sets the data range of chart series - start row, start column, end row,
end column
seriesFeb.Values = chart.ChartData[2, 3, 4, 3];
//Creates a new chart series with the name
IOfficeChartSerie seriesMarch = chart.Series.Add("March");
//Sets the data range of chart series - start row, start column, end row,
end column
seriesMarch.Values = chart.ChartData[2, 4, 4, 4];
//Sets the data range of the category axis
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData[2, 1, 4, 1];
```

```
//Specifies the chart type
chart.ChartType = OfficeChartType.Column_Clustered;
//Adds the third slide into the Presentation
pptxDoc.Save("sample.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Creates a Presentation instance
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds a blank slide to the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds chart to the slide with position and size
Dim chart As IPresentationChart = slide.Charts.AddChart(100, 10, 700, 500)
'Specifies the chart title
chart.ChartTitle = "Sales Analysis"
'Sets chart data - Row1
chart.ChartData.SetValue(1, 2, "Jan")
chart.ChartData.SetValue(1, 3, "Feb")
chart.ChartData.SetValue(1, 4, "March")
'Sets chart data - Row2
chart.ChartData.SetValue(2, 1, 2010)
chart.ChartData.SetValue(2, 2, 60)
chart.ChartData.SetValue(2, 3, 70)
chart.ChartData.SetValue(2, 4, 80)
'Sets chart data - Row3
chart.ChartData.SetValue(3, 1, 2011)
chart.ChartData.SetValue(3, 2, 80)
chart.ChartData.SetValue(3, 3, 70)
chart.ChartData.SetValue(3, 4, 60)
'Sets chart data - Row4
chart.ChartData.SetValue(4, 1, 2012)
chart.ChartData.SetValue(4, 2, 60)
chart.ChartData.SetValue(4, 3, 70)
chart.ChartData.SetValue(4, 4, 80)
'Creates a new chart series with the name
Dim seriesJan As IOfficeChartSerie = chart.Series.Add("Jan")
'Sets the data range of chart series - start row, start column, end row, end
column
seriesJan.Values = chart.ChartData(2, 2, 4, 2)
'Creates a new chart series with the name
Dim seriesFeb As IOfficeChartSerie = chart.Series.Add("Feb")
'Sets the data range of chart series - start row, start column, end row, end
column
seriesFeb.Values = chart.ChartData(2, 3, 4, 3)
'Creates a new chart series with the name
Dim seriesMarch As IOfficeChartSerie = chart.Series.Add("March")
'Sets the data range of chart series - start row, start column, end row, end
column
seriesMarch.Values = chart.ChartData(2, 4, 4, 4)
'Sets the data range of the category axis
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData(2, 1, 4, 1)
'Specifies the chart type
chart.ChartType = OfficeChartType.Column_Clustered
'Adds the third slide into the Presentation
```

```
pptxDoc.Save("sample.pptx")
'Closes the Presentation
pptxDoc.Close()
```

UWP

```
//Creates a Presentation instance
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds chart to the slide with position and size
IPresentationChart chart = slide.Charts.AddChart(100, 10, 700, 500);
//Specifies the chart title
chart.ChartTitle = "Sales Analysis";
//Sets chart data - Row1
chart.ChartData.SetValue(1, 2, "Jan");
chart.ChartData.SetValue(1, 3, "Feb");
chart.ChartData.SetValue(1, 4, "March");
//Sets chart data - Row2
chart.ChartData.SetValue(2, 1, 2010);
chart.ChartData.SetValue(2, 2, 60);
chart.ChartData.SetValue(2, 3, 70);
chart.ChartData.SetValue(2, 4, 80);
//Sets chart data - Row3
chart.ChartData.SetValue(3, 1, 2011);
chart.ChartData.SetValue(3, 2, 80);
chart.ChartData.SetValue(3, 3, 70);
chart.ChartData.SetValue(3, 4, 60);
//Sets chart data - Row4
chart.ChartData.SetValue(4, 1, 2012);
chart.ChartData.SetValue(4, 2, 60);
chart.ChartData.SetValue(4, 3, 70);
chart.ChartData.SetValue(4, 4, 80);
//Creates a new chart series with the name
IOfficeChartSerie seriesJan = chart.Series.Add("Jan");
//Sets the data range of chart series - start row, start column, end row,
end column
seriesJan.Values = chart.ChartData[2, 2, 4, 2];
//Creates a new chart series with the name
IOfficeChartSerie seriesFeb = chart.Series.Add("Feb");
//Sets the data range of chart series - start row, start column, end row,
end column
seriesFeb.Values = chart.ChartData[2, 3, 4, 3];
//Creates a new chart series with the name
IOfficeChartSerie seriesMarch = chart.Series.Add("March");
//Sets the data range of chart series - start row, start column, end row,
end column
seriesMarch.Values = chart.ChartData[2, 4, 4, 4];
//Sets the data range of the category axis
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData[2, 1, 4, 1];
//Specifies the chart type
chart.ChartType = OfficeChartType.Column_Clustered;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "sample";
```



```

savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates a Presentation instance
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds chart to the slide with position and size
IPresentationChart chart = slide.Charts.AddChart(100, 10, 700, 500);
//Specifies the chart title
chart.ChartTitle = "Sales Analysis";
//Sets chart data - Row1
chart.ChartData.SetValue(1, 2, "Jan");
chart.ChartData.SetValue(1, 3, "Feb");
chart.ChartData.SetValue(1, 4, "March");
//Sets chart data - Row2
chart.ChartData.SetValue(2, 1, 2010);
chart.ChartData.SetValue(2, 2, 60);
chart.ChartData.SetValue(2, 3, 70);
chart.ChartData.SetValue(2, 4, 80);
//Sets chart data - Row3
chart.ChartData.SetValue(3, 1, 2011);
chart.ChartData.SetValue(3, 2, 80);
chart.ChartData.SetValue(3, 3, 70);
chart.ChartData.SetValue(3, 4, 60);
//Sets chart data - Row4
chart.ChartData.SetValue(4, 1, 2012);
chart.ChartData.SetValue(4, 2, 60);
chart.ChartData.SetValue(4, 3, 70);
chart.ChartData.SetValue(4, 4, 80);
//Creates a new chart series with the name
IOfficeChartSerie seriesJan = chart.Series.Add("Jan");
//Sets the data range of chart series - start row, start column, end row, end column
seriesJan.Values = chart.ChartData[2, 2, 4, 2];
//Creates a new chart series with the name
IOfficeChartSerie seriesFeb = chart.Series.Add("Feb");
//Sets the data range of chart series - start row, start column, end row, end column
seriesFeb.Values = chart.ChartData[2, 3, 4, 3];
//Creates a new chart series with the name
IOfficeChartSerie seriesMarch = chart.Series.Add("March");
//Sets the data range of chart series - start row, start column, end row, end column
seriesMarch.Values = chart.ChartData[2, 4, 4, 4];
//Sets the data range of the category axis
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData[2, 1, 4, 1];
//Specifies the chart type
chart.ChartType = OfficeChartType.Column_Clustered;
//Save the PowerPoint Presentation as stream

```

```

FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

//Creates a Presentation instance
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds chart to the slide with position and size
IPresentationChart chart = slide.Charts.AddChart(100, 10, 700, 500);
//Specifies the chart title
chart.ChartTitle = "Sales Analysis";
//Sets chart data - Row1
chart.ChartData.SetValue(1, 2, "Jan");
chart.ChartData.SetValue(1, 3, "Feb");
chart.ChartData.SetValue(1, 4, "March");
//Sets chart data - Row2
chart.ChartData.SetValue(2, 1, 2010);
chart.ChartData.SetValue(2, 2, 60);
chart.ChartData.SetValue(2, 3, 70);
chart.ChartData.SetValue(2, 4, 80);
//Sets chart data - Row3
chart.ChartData.SetValue(3, 1, 2011);
chart.ChartData.SetValue(3, 2, 80);
chart.ChartData.SetValue(3, 3, 70);
chart.ChartData.SetValue(3, 4, 60);
//Sets chart data - Row4
chart.ChartData.SetValue(4, 1, 2012);
chart.ChartData.SetValue(4, 2, 60);
chart.ChartData.SetValue(4, 3, 70);
chart.ChartData.SetValue(4, 4, 80);
//Creates a new chart series with the name
IOfficeChartSerie seriesJan = chart.Series.Add("Jan");
//Sets the data range of chart series - start row, start column, end row,
end column
seriesJan.Values = chart.ChartData[2, 2, 4, 2];
//Creates a new chart series with the name
IOfficeChartSerie seriesFeb = chart.Series.Add("Feb");
//Sets the data range of chart series - start row, start column, end row,
end column
seriesFeb.Values = chart.ChartData[2, 3, 4, 3];
//Creates a new chart series with the name
IOfficeChartSerie seriesMarch = chart.Series.Add("March");
//Sets the data range of chart series - start row, start column, end row,
end column
seriesMarch.Values = chart.ChartData[2, 4, 4, 4];
//Sets the data range of the category axis
chart.PrimaryCategoryAxis.CategoryLabels = chart.ChartData[2, 1, 4, 1];
//Specifies the chart type
chart.ChartType = OfficeChartType.Column_Clustered;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.

```

```
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Creating charts from excel sheet

You can also create a chart with the data from an existing excel worksheet. The following code example demonstrates the same.

C#

```
//Creates a Presentation instance
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Gets the excel file as stream
MemoryStream excelStream = new
MemoryStream(File.ReadAllBytes("Book1.xlsx"));
//Adds a chart to the slide with a data range from excel worksheet - excel
workbook, worksheet number, Data range, position, and size.
IPresentationChart chart = slide.Charts.AddChart(excelStream, 1, "A1:D4",
new RectangleF(100, 10, 700, 500));
//Saves the Presentation
pptxDoc.Save("output.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Creates a Presentation instance
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds a blank slide to the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Gets the excel file as stream
Dim excelStream As New MemoryStream(File.ReadAllBytes("Book1.xlsx"))
'Adds a chart to the slide with a data range from excel worksheet - excel
workbook, worksheet number, Data range, position, and size.
Dim chart As IPresentationChart = slide.Charts.AddChart(excelStream, 1,
"A1:D4", New RectangleF(100, 10, 700, 500))
'Saves the Presentation
pptxDoc.Save("output.pptx")
'Closes the Presentation
pptxDoc.Close()
```

UWP

```

//Creates a Presentation instance
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Gets the excel file as stream
MemoryStream excelStream = new
MemoryStream(File.ReadAllBytes("Book1.xlsx"));
//Adds a chart to the slide with a data range from excel worksheet - excel
workbook, worksheet number, Data range, position, and size.
IPresentationChart chart = slide.Charts.AddChart(excelStream, 1, "A1:D4",
new RectangleF(100, 10, 700, 500));
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates a Presentation instance
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Gets the excel file as stream
FileStream excelStream = new FileStream("Book1.xlsx", FileMode.Open);
//Adds a chart to the slide with a data range from excel worksheet - excel
workbook, worksheet number, Data range, position, and size.
IPresentationChart chart = slide.Charts.AddChart(excelStream, 1, "A1:D4",
new RectangleF(100, 10, 700, 500));
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

//Creates a Presentation instance
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets the excel file as stream
Stream excelStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Book1.xlsx");

```

```

//Adds a chart to the slide with a data range from excel worksheet - excel
workbook, worksheet number, Data range, position, and size.
IPresentationChart chart = slide.Charts.AddChart(excelStream, 1, "A1:D4",
new RectangleF(100, 10, 700, 500));
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Creating Custom Charts

Essential Presentation facilitates you to create custom charts by adding different charts series for a single chart.

For example, you can use a Bar- clustered chart for the first data series and a scatter- line- marker chart for the second series. As a result, you can have a Bar-clustered chart combined with a scatter-line-marker chart.

The following code example demonstrates how to create custom charts.

C#

```

//Creates an instance of the IPresentation
IPresentation pptxDoc = Presentation.Create();
//Creates a new slide
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a new chart in the slide by specifying its position and size as
parameters.
IPresentationChart chart = slide.Charts.AddChart(100, 80, 500, 350);
chart.ChartTitle = "Sales comparison";
chart.ChartTitleArea.Bold = true;
//Sets the data for chart- RowIndex, columnIndex and data
chart.ChartData.SetValue(1, 1, "Month");
chart.ChartData.SetValue(2, 1, "July");
chart.ChartData.SetValue(3, 1, "August");
chart.ChartData.SetValue(4, 1, "September");
chart.ChartData.SetValue(5, 1, "October");
chart.ChartData.SetValue(6, 1, "November");
chart.ChartData.SetValue(7, 1, "December");
chart.ChartData.SetValue(1, 2, "2013");
chart.ChartData.SetValue(2, 2, 35);
chart.ChartData.SetValue(3, 2, 47);

```

```

chart.ChartData.SetValue(4, 2, 30);
chart.ChartData.SetValue(5, 2, 29);
chart.ChartData.SetValue(6, 2, 25);
chart.ChartData.SetValue(7, 2, 30);
chart.ChartData.SetValue(1, 3, "2014");
chart.ChartData.SetValue(2, 3, 30);
chart.ChartData.SetValue(3, 3, 25);
chart.ChartData.SetValue(4, 3, 29);
chart.ChartData.SetValue(5, 3, 35);
chart.ChartData.SetValue(6, 3, 38);
chart.ChartData.SetValue(7, 3, 32);
chart.ChartData.SetValue(1, 4, "2015");
chart.ChartData.SetValue(2, 4, 35);
chart.ChartData.SetValue(3, 4, 37);
chart.ChartData.SetValue(4, 4, 30);
chart.ChartData.SetValue(5, 4, 29);
chart.ChartData.SetValue(6, 4, 25);
chart.ChartData.SetValue(7, 4, 30);
//Creates a new ChartSerie with the name
IOfficeChartSerie serie2013 = chart.Series.Add("2013");
//Sets the data range of chart series start row, start column, end row, end column
serie2013.Values = chart.ChartData[2, 2, 7, 2];
serie2013.SeriesType = OfficeChartType.Bar_Clustered;
IOfficeChartSerie serie2014 = chart.Series.Add("2014");
serie2014.Values = chart.ChartData[2, 3, 7, 3];
serie2014.SeriesType = OfficeChartType.Scatter_Line_Markers;
//Saves the Presentation
pptxDoc.Save("Output_1.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Creates an instance of the IPresentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Creates a new slide
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds a new chart in the slide by specifying its position and size as parameters.
Dim chart As IPresentationChart = slide.Charts.AddChart(100, 80, 500, 350)
chart.ChartTitle = "Sales comparison"
chart.ChartTitleArea.Bold = True
'Sets the data for chart- RowIndex, columnIndex, and data
chart.ChartData.SetValue(1, 1, "Month")
chart.ChartData.SetValue(2, 1, "July")
chart.ChartData.SetValue(3, 1, "August")
chart.ChartData.SetValue(4, 1, "September")
chart.ChartData.SetValue(5, 1, "October")
chart.ChartData.SetValue(6, 1, "November")
chart.ChartData.SetValue(7, 1, "December")
chart.ChartData.SetValue(1, 2, "2013")
chart.ChartData.SetValue(2, 2, 35)
chart.ChartData.SetValue(3, 2, 47)
chart.ChartData.SetValue(4, 2, 30)
chart.ChartData.SetValue(5, 2, 29)

```

```

chart.ChartData.SetValue(6, 2, 25)
chart.ChartData.SetValue(7, 2, 30)
chart.ChartData.SetValue(1, 3, "2014")
chart.ChartData.SetValue(2, 3, 30)
chart.ChartData.SetValue(3, 3, 25)
chart.ChartData.SetValue(4, 3, 29)
chart.ChartData.SetValue(5, 3, 35)
chart.ChartData.SetValue(6, 3, 38)
chart.ChartData.SetValue(7, 3, 32)
chart.ChartData.SetValue(1, 4, "2015")
chart.ChartData.SetValue(2, 4, 35)
chart.ChartData.SetValue(3, 4, 37)
chart.ChartData.SetValue(4, 4, 30)
chart.ChartData.SetValue(5, 4, 29)
chart.ChartData.SetValue(6, 4, 25)
chart.ChartData.SetValue(7, 4, 30)
'Creates a new ChartSerie with the name
Dim serie2013 As IOOfficeChartSerie = chart.Series.Add("2013")
'Sets the data range of chart series start row, start column, end row, end
column
serie2013.Values = chart.ChartData(2, 2, 7, 2)
serie2013.SeriesType = OfficeChartType.Bar_Clustered
Dim serie2014 As IOOfficeChartSerie = chart.Series.Add("2014")
serie2014.Values = chart.ChartData(2, 3, 7, 3)
serie2014.SeriesType = OfficeChartType.Scatter_Line_Markers
'Saves the Presentation
pptxDoc.Save("Output_1.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Creates an instance of the IPresentation
IPresentation pptxDoc = Presentation.Create();
//Creates a new slide
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a new chart in the slide by specifying its position and size as
parameters.
IPresentationChart chart = slide.Charts.AddChart(100, 80, 500, 350);
chart.ChartTitle = "Sales comparison";
chart.ChartTitleArea.Bold = true;
//Sets the data for chart- RowIndex, columnIndex and data
chart.ChartData.SetValue(1, 1, "Month");
chart.ChartData.SetValue(2, 1, "July");
chart.ChartData.SetValue(3, 1, "August");
chart.ChartData.SetValue(4, 1, "September");
chart.ChartData.SetValue(5, 1, "October");
chart.ChartData.SetValue(6, 1, "November");
chart.ChartData.SetValue(7, 1, "December");
chart.ChartData.SetValue(1, 2, "2013");
chart.ChartData.SetValue(2, 2, 35);
chart.ChartData.SetValue(3, 2, 47);
chart.ChartData.SetValue(4, 2, 30);
chart.ChartData.SetValue(5, 2, 29);
chart.ChartData.SetValue(6, 2, 25);
chart.ChartData.SetValue(7, 2, 30);

```

```

chart.ChartData.SetValue(1, 3, "2014");
chart.ChartData.SetValue(2, 3, 30);
chart.ChartData.SetValue(3, 3, 25);
chart.ChartData.SetValue(4, 3, 29);
chart.ChartData.SetValue(5, 3, 35);
chart.ChartData.SetValue(6, 3, 38);
chart.ChartData.SetValue(7, 3, 32);
chart.ChartData.SetValue(1, 4, "2015");
chart.ChartData.SetValue(2, 4, 35);
chart.ChartData.SetValue(3, 4, 37);
chart.ChartData.SetValue(4, 4, 30);
chart.ChartData.SetValue(5, 4, 29);
chart.ChartData.SetValue(6, 4, 25);
chart.ChartData.SetValue(7, 4, 30);
//Creates a new ChartSerie with the name
IOfficeChartSerie serie2013 = chart.Series.Add("2013");
//Sets the data range of chart series start row, start column, end row, end
column
serie2013.Values = chart.ChartData[2, 2, 7, 2];
serie2013.SerieType = OfficeChartType.Bar_Clustered;
IOfficeChartSerie serie2014 = chart.Series.Add("2014");
serie2014.Values = chart.ChartData[2, 3, 7, 3];
serie2014.SerieType = OfficeChartType.Scatter_Line_Markers;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output_1";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates an instance of the IPresentation
IPresentation pptxDoc = Presentation.Create();
//Creates a new slide
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a new chart in the slide by specifying its position and size as
parameters.
IPresentationChart chart = slide.Charts.AddChart(100, 80, 500, 350);
chart.ChartTitle = "Sales comparison";
chart.ChartTitleArea.Bold = true;
//Sets the data for chart- RowIndex, columnIndex and data
chart.ChartData.SetValue(1, 1, "Month");
chart.ChartData.SetValue(2, 1, "July");
chart.ChartData.SetValue(3, 1, "August");
chart.ChartData.SetValue(4, 1, "September");
chart.ChartData.SetValue(5, 1, "October");
chart.ChartData.SetValue(6, 1, "November");
chart.ChartData.SetValue(7, 1, "December");
chart.ChartData.SetValue(1, 2, "2013");
chart.ChartData.SetValue(2, 2, 35);
chart.ChartData.SetValue(3, 2, 47);

```



```

chart.ChartData.SetValue(4, 2, 30);
chart.ChartData.SetValue(5, 2, 29);
chart.ChartData.SetValue(6, 2, 25);
chart.ChartData.SetValue(7, 2, 30);
chart.ChartData.SetValue(1, 3, "2014");
chart.ChartData.SetValue(2, 3, 30);
chart.ChartData.SetValue(3, 3, 25);
chart.ChartData.SetValue(4, 3, 29);
chart.ChartData.SetValue(5, 3, 35);
chart.ChartData.SetValue(6, 3, 38);
chart.ChartData.SetValue(7, 3, 32);
chart.ChartData.SetValue(1, 4, "2015");
chart.ChartData.SetValue(2, 4, 35);
chart.ChartData.SetValue(3, 4, 37);
chart.ChartData.SetValue(4, 4, 30);
chart.ChartData.SetValue(5, 4, 29);
chart.ChartData.SetValue(6, 4, 25);
chart.ChartData.SetValue(7, 4, 30);
//Creates a new ChartSeries with the name
IOfficeChartSerie serie2013 = chart.Series.Add("2013");
//Sets the data range of chart series start row, start column, end row, end
column
serie2013.Values = chart.ChartData[2, 2, 7, 2];
serie2013.SeriesType = OfficeChartType.Bar_Clustered;
IOfficeChartSerie serie2014 = chart.Series.Add("2014");
serie2014.Values = chart.ChartData[2, 3, 7, 3];
serie2014.SeriesType = OfficeChartType.Scatter_Line_Markers;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output_1.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

//Creates an instance of the IPresentation
IPresentation pptxDoc = Presentation.Create();
//Creates a new slide
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a new chart in the slide by specifying its position and size as
parameters.
IPresentationChart chart = slide.Charts.AddChart(100, 80, 500, 350);
chart.ChartTitle = "Sales comparison";
chart.ChartTitleArea.Bold = true;
//Sets the data for chart- RowIndex, columnIndex and data
chart.ChartData.SetValue(1, 1, "Month");
chart.ChartData.SetValue(2, 1, "July");
chart.ChartData.SetValue(3, 1, "August");
chart.ChartData.SetValue(4, 1, "September");
chart.ChartData.SetValue(5, 1, "October");
chart.ChartData.SetValue(6, 1, "November");
chart.ChartData.SetValue(7, 1, "December");
chart.ChartData.SetValue(1, 2, "2013");
chart.ChartData.SetValue(2, 2, 35);
chart.ChartData.SetValue(3, 2, 47);
chart.ChartData.SetValue(4, 2, 30);

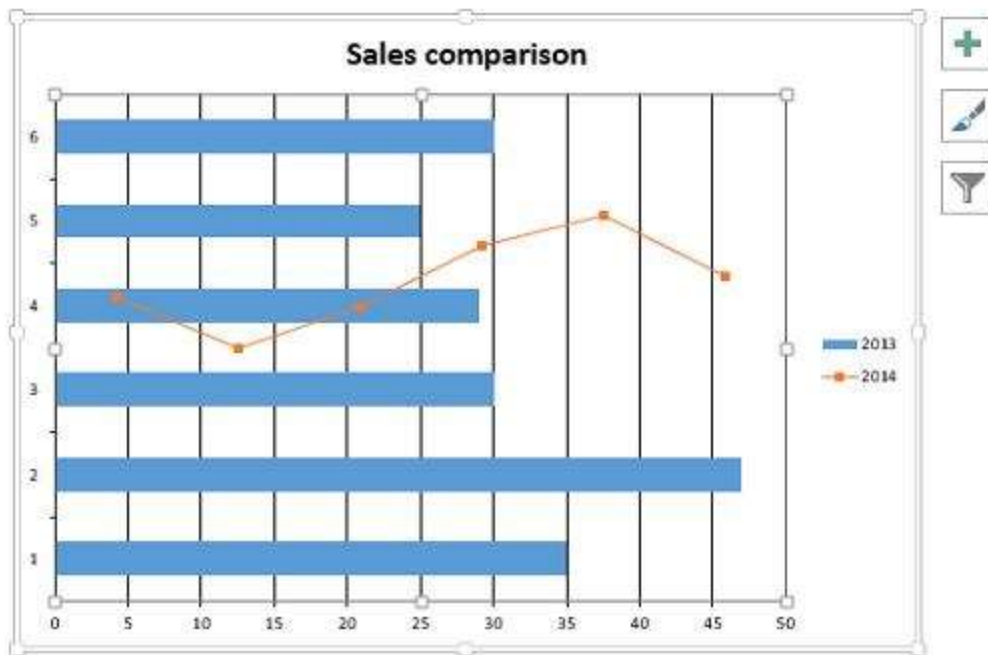
```

```

chart.ChartData.SetValue(5, 2, 29);
chart.ChartData.SetValue(6, 2, 25);
chart.ChartData.SetValue(7, 2, 30);
chart.ChartData.SetValue(1, 3, "2014");
chart.ChartData.SetValue(2, 3, 30);
chart.ChartData.SetValue(3, 3, 25);
chart.ChartData.SetValue(4, 3, 29);
chart.ChartData.SetValue(5, 3, 35);
chart.ChartData.SetValue(6, 3, 38);
chart.ChartData.SetValue(7, 3, 32);
chart.ChartData.SetValue(1, 4, "2015");
chart.ChartData.SetValue(2, 4, 35);
chart.ChartData.SetValue(3, 4, 37);
chart.ChartData.SetValue(4, 4, 30);
chart.ChartData.SetValue(5, 4, 29);
chart.ChartData.SetValue(6, 4, 25);
chart.ChartData.SetValue(7, 4, 30);
//Creates a new ChartSerie with the name
IOfficeChartSerie serie2013 = chart.Series.Add("2013");
//Sets the data range of chart series start row, start column, end row, end
column
serie2013.Values = chart.ChartData[2, 2, 7, 2];
serie2013.SerieType = OfficeChartType.Bar_Clustered;
IOfficeChartSerie serie2014 = chart.Series.Add("2014");
serie2014.Values = chart.ChartData[2, 3, 7, 3];
serie2014.SerieType = OfficeChartType.Scatter_Line_Markers;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output_1.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output_1.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

The above code example creates a chart in the following screenshot.



Refreshing the chart

Sometimes, the charts do not represent the actual data. In those cases, the charts in PowerPoint Presentation should be refreshed.

The following code example demonstrates how to refresh the charts in PowerPoint Presentation.

C#

```
//Opens the Presentation
IPresentation pptxDoc = Presentation.Open("Chart.pptx");
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Refreshes the chart
chart.Refresh();
//Saves the Presentation
pptxDoc.Save("output.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Opens the Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Chart.pptx")
'Gets the first slide
Dim slide As ISlide = pptxDoc.Slides(0)
'Gets the chart in slide
Dim chart As IPresentationChart = TryCast(slide.Shapes(0),
IPresentationChart)
'Refreshes the chart
chart.Refresh()
```

```
'Saves the Presentation
pptxDoc.Save("output.pptx")
'Closes the Presentation
pptxDoc.Close()
```

UWP

```
//Opens the Presentation
IPresentation pptxDoc = Presentation.Open("Chart.pptx");
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Refreshes the chart
chart.Refresh();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Refreshes the chart
chart.Refresh();
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```
/"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
```

```

IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Refreshes the chart
chart.Refresh();
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Editing the Chart Data

You can change the data for an existing chart. The code example demonstrates how to modify the chart in a slide.

C#

```

//Opens a Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Adds a slide to the Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Modifies chart data - Row1
chart.ChartData.SetValue(1, 2, "Jan");
chart.ChartData.SetValue(1, 3, "Feb");
chart.ChartData.SetValue(1, 4, "March");
//Modifies chart data - Row2
chart.ChartData.SetValue(2, 1, 2010);
chart.ChartData.SetValue(2, 2, 60);
chart.ChartData.SetValue(2, 3, 70);
chart.ChartData.SetValue(2, 4, 80);
//Refreshes the chart
chart.Refresh();
//Saves the Presentation
pptxDoc.Save("output.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Opens a Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")

```

```

'Adds a slide to the Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Gets the chart in slide
Dim chart As IPresentationChart = TryCast(slide.Shapes(0),
IPresentationChart)
'Modifies chart data - Row1
chart.ChartData.SetValue(1, 2, "Jan")
chart.ChartData.SetValue(1, 3, "Feb")
chart.ChartData.SetValue(1, 4, "March")
'Modifies chart data - Row2
chart.ChartData.SetValue(2, 1, 2010)
chart.ChartData.SetValue(2, 2, 60)
chart.ChartData.SetValue(2, 3, 70)
chart.ChartData.SetValue(2, 4, 80)
'Refreshes the chart
chart.Refresh()
'Saves the Presentation
pptxDoc.Save("output.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Opens a Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Adds a slide to the Presentation
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Modifies chart data - Row1
chart.ChartData.SetValue(1, 2, "Jan");
chart.ChartData.SetValue(1, 3, "Feb");
chart.ChartData.SetValue(1, 4, "March");
//Modifies chart data - Row2
chart.ChartData.SetValue(2, 1, 2010);
chart.ChartData.SetValue(2, 2, 60);
chart.ChartData.SetValue(2, 3, 70);
chart.ChartData.SetValue(2, 4, 80);
//Refreshes the chart
chart.Refresh();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);

```

```
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Modifies chart data - Row1
chart.ChartData.SetValue(1, 2, "Jan");
chart.ChartData.SetValue(1, 3, "Feb");
chart.ChartData.SetValue(1, 4, "March");
//Modifies chart data - Row2
chart.ChartData.SetValue(2, 1, 2010);
chart.ChartData.SetValue(2, 2, 60);
chart.ChartData.SetValue(2, 3, 70);
chart.ChartData.SetValue(2, 4, 80);
//Refreshes the chart
chart.Refresh();
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```
/"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Modifies chart data - Row1
chart.ChartData.SetValue(1, 2, "Jan");
chart.ChartData.SetValue(1, 3, "Feb");
chart.ChartData.SetValue(1, 4, "March");
//Modifies chart data - Row2
chart.ChartData.SetValue(2, 1, 2010);
chart.ChartData.SetValue(2, 2, 60);
chart.ChartData.SetValue(2, 3, 70);
chart.ChartData.SetValue(2, 4, 80);
//Refreshes the chart
chart.Refresh();
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.
```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Customizing the chart

Chart Basics

A chart is composed of various elements such as legends, axes, series, etc. Each chart element corresponds to an object. The following image illustrates the basic elements of a chart.



1. The chart area of the chart.
2. The plot area of the chart.
3. The data points of the data series that are plotted in the chart.
4. The horizontal (category) and vertical (value) axis along where the data is plotted in the chart.
5. The legend of the chart.
6. A chart and axis title that you can use in the chart.
7. A data label that you can use to identify the details of a data point in a data series.

Modifying the Chart Appearance

The appearance of a chart can be modified according to the convenience and usage. The following code example demonstrates modifying the chart element styles.

C#

```

//Opens the Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide

```



```

IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Modifies the chart height
chart.Height = 500;
//Modifies the chart width
chart.Width = 700;
//Changes the title
chart.ChartTitle = "New title";
//Changes the series name of first chart series
chart.Series[0].Name = "Modified series name";
//Hides the category labels
chart.CategoryLabelLevel =
OfficeCategoriesLabelLevel.CategoriesLabelLevelNone;
//Shows Data Table.
chart.HasDataTable = true;
//Formats Chart Area.
IOfficeChartFrameFormat chartArea = chart.ChartArea;
//Chart Area Border Settings
//Style
chartArea.Border.LinePattern = OfficeChartLinePattern.Solid;
//Color
chartArea.Border.LineColor = Color.Blue;
//Weight
chartArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Chart Area Settings
//Fill Effects
chartArea.Fill.FillType = OfficeFillType.Gradient;
//Two Color
chartArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
//Sets two colors.
chartArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartArea.Fill.ForeColor = Color.White;
//Plot Area
IOfficeChartFrameFormat chartPlotArea = chart.PlotArea;
//Plots Area Border Settings
//Style
chartPlotArea.Border.LinePattern = OfficeChartLinePattern.Solid;
//Color
chartPlotArea.Border.LineColor = Color.Blue;
//Weight
chartPlotArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Fill Effects
chartPlotArea.Fill.FillType = OfficeFillType.Gradient;
//Two Color
chartPlotArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
//Sets two colors.
chartPlotArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartPlotArea.Fill.ForeColor = Color.White;
//Saves the Presentation
pptxDoc.Save("Output.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Opens the Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")

```

```

'Gets the first slide
Dim slide As ISlide = pptxDoc.Slides(0)
'Gets the chart in slide
Dim chart__2 As IPresentationChart = TryCast(slide.Shapes(0),
IPresentationChart)
'Modifies the chart height
chart__2.Height = 500
'Modifies the chart width
chart__2.Width = 700
'Changes the title
chart__2.ChartTitle = "New title"
'Changes the series name of first chart series
chart__2.Series(0).Name = "Modified series name"
'Hides the category labels
chart__2.CategoryLabelLevel =
OfficeCategoriesLabelLevel.CategoriesLabelLevelNone
'Shows Data Table.
chart__2.HasDataTable = True
'Formats Chart Area.
Dim chartArea As IOfficeChartFrameFormat = chart__2.ChartArea
'Chart Area Border Settings
'Style
chartArea.Border.LinePattern = OfficeChartLinePattern.Solid
'Color
chartArea.Border.LineColor = Color.Blue
'Weight
chartArea.Border.LineWeight = OfficeChartLineWeight.Hairline
'Chart Area Settings
'Fill Effects
chartArea.Fill.FillType = OfficeFillType.Gradient
'Two Color
chartArea.Fill.GradientColorType = OfficeGradientColor.TwoColor
'Sets two colors.
chartArea.Fill.BackColor = Color.FromArgb(205, 217, 234)
chartArea.Fill.ForeColor = Color.White
'Plots Area
Dim chartPlotArea As IOfficeChartFrameFormat = chart__2.PlotArea
'Plots Area Border Settings
'Style
chartPlotArea.Border.LinePattern = OfficeChartLinePattern.Solid
'Color
chartPlotArea.Border.LineColor = Color.Blue
'Weight
chartPlotArea.Border.LineWeight = OfficeChartLineWeight.Hairline
'Fill Effects
chartPlotArea.Fill.FillType = OfficeFillType.Gradient
'Two Color
chartPlotArea.Fill.GradientColorType = OfficeGradientColor.TwoColor
'Sets two colors.
chartPlotArea.Fill.BackColor = Color.FromArgb(205, 217, 234)
chartPlotArea.Fill.ForeColor = Color.White
'Saves the Presentation
pptxDoc.Save("Output.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```
//Opens the Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Modifies the chart height
chart.Height = 500;
//Modifies the chart width
chart.Width = 700;
//Changes the title
chart.ChartTitle = "New title";
//Changes the series name of first chart series
chart.Series[0].Name = "Modified series name";
//Hides the category labels
chart.CategoryLabelLevel =
OfficeCategoriesLabelLevel.CategoriesLabelLevelNone;
//Shows Data Table.
chart.HasDataTable = true;
//Formats Chart Area.
IOfficeChartFrameFormat chartArea = chart.ChartArea;
//Chart Area Border Settings
//Style
chartArea.Border.LinePattern = OfficeChartLinePattern.Solid;
//Color
chartArea.Border.LineColor = Color.Blue;
//Weight
chartArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Chart Area Settings
//Fill Effects
chartArea.Fill.FillType = OfficeFillType.Gradient;
//Two Color
chartArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
//Sets two colors.
chartArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartArea.Fill.ForeColor = Color.White;
//Plot Area
IOfficeChartFrameFormat chartPlotArea = chart.PlotArea;
//Plots Area Border Settings
//Style
chartPlotArea.Border.LinePattern = OfficeChartLinePattern.Solid;
//Color
chartPlotArea.Border.LineColor = Color.Blue;
//Weight
chartPlotArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Fill Effects
chartPlotArea.Fill.FillType = OfficeFillType.Gradient;
//Two Color
chartPlotArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
//Sets two colors.
chartPlotArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartPlotArea.Fill.ForeColor = Color.White;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
```

```

savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Modifies the chart height
chart.Height = 500;
//Modifies the chart width
chart.Width = 700;
//Changes the title
chart.ChartTitle = "New title";
//Changes the series name of first chart series
chart.Series[0].Name = "Modified series name";
//Hides the category labels
chart.CategoryLabelLevel =
OfficeCategoriesLabelLevel.CategoriesLabelLevelNone;
//Shows Data Table.
chart.HasDataTable = true;
//Formats Chart Area.
IOfficeChartFrameFormat chartArea = chart.ChartArea;
//Chart Area Border Settings
//Style
chartArea.Border.LinePattern = OfficeChartLinePattern.Solid;
//Color
chartArea.Border.LineColor = Color.Blue;
//Weight
chartArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Chart Area Settings
//Fill Effects
chartArea.Fill.FillType = OfficeFillType.Gradient;
//Two Color
chartArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
//Sets two colors.
chartArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartArea.Fill.ForeColor = Color.White;
//Plot Area
IOfficeChartFrameFormat chartPlotArea = chart.PlotArea;
//Plots Area Border Settings
//Style
chartPlotArea.Border.LinePattern = OfficeChartLinePattern.Solid;
//Color
chartPlotArea.Border.LineColor = Color.Blue;
//Weight
chartPlotArea.Border.LineWeight = OfficeChartLineWeight.Hairline;

```

```
//Fill Effects
chartPlotArea.Fill.FillType = OfficeFillType.Gradient;
//Two Color
chartPlotArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
//Sets two colors.
chartPlotArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartPlotArea.Fill.ForeColor = Color.White;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Modifies the chart height
chart.Height = 500;
//Modifies the chart width
chart.Width = 700;
//Changes the title
chart.ChartTitle = "New title";
//Changes the series name of first chart series
chart.Series[0].Name = "Modified series name";
//Hides the category labels
chart.CategoryLabelLevel =
OfficeCategoriesLabelLevel.CategoriesLabelLevelNone;
//Formats Chart Area.
IOfficeChartFrameFormat chartArea = chart.ChartArea;
//Chart Area Border Settings
//Style
chartArea.Border.LinePattern = OfficeChartLinePattern.Solid;
//Color
chartArea.Border.LineColor = Color.Blue;
//Weight
chartArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Chart Area Settings
//Fill Effects
chartArea.Fill.FillType = OfficeFillType.Gradient;
//Two Color
chartArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
//Sets two colors.
chartArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartArea.Fill.ForeColor = Color.White;
//Plot Area
IOfficeChartFrameFormat chartPlotArea = chart.PlotArea;
```

```

//Plots Area Border Settings
//Style
chartPlotArea.Border.LinePattern = OfficeChartLinePattern.Solid;
//Color
chartPlotArea.Border.LineColor = Color.Blue;
//Weight
chartPlotArea.Border.LineWeight = OfficeChartLineWeight.Hairline;
//Fill Effects
chartPlotArea.Fill.FillType = OfficeFillType.Gradient;
//Two Color
chartPlotArea.Fill.GradientColorType = OfficeGradientColor.TwoColor;
//Sets two colors.
chartPlotArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartPlotArea.Fill.ForeColor = Color.White;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Modifying the Plot and Legends of chart

The following code example demonstrates how to modify the legend and plot areas of a chart.

C#

```

//Opens a Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Gets the chart from the first slide
IPresentationChart chart = pptxDoc.Slides[0].Charts[0] as
IPresentationChart;
//Sets border settings
chart.PlotArea.Border.AutoFormat = false;
//Sets the auto line color
chart.PlotArea.Border.IsAutoLineColor = false;
//Sets the border line color
chart.PlotArea.Border.LineColor = Color.Blue;
//Sets the border line pattern
chart.PlotArea.Border.LinePattern = OfficeChartLinePattern.DashDot;
//Sets the border line weight
chart.PlotArea.Border.LineWeight = OfficeChartLineWeight.Wide;
//Sets the border transparency
chart.PlotArea.Border.Transparency = 0.6;

```

```
//Sets the plot area's fill type
chart.PlotArea.Fill.FillType = OfficeFillType.SolidColor;
//Sets the plot area's fill color
chart.PlotArea.Fill.ForeColor = Color.LightPink;
//Sets the plot area shadow presence
chart.PlotArea.Shadow.ShadowInnerPresets =
Office2007ChartPresetsInner.InsideDiagonalTopLeft;
//Sets the legend position
chart.Legend.Position = OfficeLegendPosition.Left;
//Sets the layout inclusion
chart.Legend.IncludeInLayout = true;
//Sets the legend border format
chart.Legend.FrameFormat.Border.AutoFormat = false;
//Sets the legend border auto line color
chart.Legend.FrameFormat.Border.IsAutoLineColor = false;
//Sets the border line color
chart.Legend.FrameFormat.Border.LineColor = Color.Blue;
//Sets the border line pattern
chart.Legend.FrameFormat.Border.LinePattern =
OfficeChartLinePattern.DashDot;
//Sets the legend border line weight
chart.Legend.FrameFormat.Border.LineWeight = OfficeChartLineWeight.Wide;
//Sets the font weight for legend text
chart.Legend.TextArea.Bold = true;
//Sets the fore color to legend text
chart.Legend.TextArea.Color = OfficeKnownColors.Bright_green;
//Sets the font name for legend text
chart.Legend.TextArea.FontName = "Times New Roman";
//Sets the font size for legend text
chart.Legend.TextArea.Size = 20;
//Sets the legend text area' strike through
chart.Legend.TextArea.Strikethrough = true;
//Modifies the legend entry
chart.Legend.LegendEntries[0].IsDeleted = true;
//Modifies the legend layout height
chart.Legend.Layout.Height = 200;
//Modifies the legend layout height mode
chart.Legend.Layout.HeightMode = LayoutModes.factor;
//Modifies the legend layout left position
chart.Legend.Layout.Left = 100;
//Modifies the legend layout left mode
chart.Legend.Layout.LeftMode = LayoutModes.factor;
//Modifies the legend layout top position
chart.Legend.Layout.Top = 100;
//Modifies the legend layout top mode
chart.Legend.Layout.TopMode = LayoutModes.factor;
//Modifies the legend layout width
chart.Legend.Layout.Width = 300;
//Modifies the legend layout width mode
chart.Legend.Layout.WidthMode = LayoutModes.factor;
//Saves the Presentation
pptxDoc.Save("ModifiedChart.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Opens a Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Gets the chart from the first slide
Dim chart As IPresentationChart = TryCast(pptxDoc.Slides(0).Charts(0),
IPresentationChart)
'Sets border settings
chart.PlotArea.Border.AutoFormat = False
'Sets the auto line color
chart.PlotArea.Border.IsAutoLineColor = False
'Sets the border line color
chart.PlotArea.Border.LineColor = Color.Blue
'Sets the border line pattern
chart.PlotArea.Border.LinePattern = OfficeChartLinePattern.DashDot
'Sets the border line weight
chart.PlotArea.Border.LineWeight = OfficeChartLineWeight.Wide
'Sets the border transparency
chart.PlotArea.Border.Transparency = 0.6
'Sets the plot area's fill type
chart.PlotArea.Fill.FillType = OfficeFillType.SolidColor
'Sets the plot area's fill color
chart.PlotArea.Fill.ForeColor = Color.LightPink
'Sets the plot area shadow presence
chart.PlotArea.Shadow.ShadowInnerPresets =
Office2007ChartPresetsInner.InsideDiagonalTopLeft
'Sets the legend position
chart.Legend.Position = OfficeLegendPosition.Left
'Sets the layout inclusion
chart.Legend.IncludeInLayout = True
'Sets the legend border format
chart.Legend.FrameFormat.Border.AutoFormat = False
'Sets the legend border auto line color
chart.Legend.FrameFormat.Border.IsAutoLineColor = False
'Sets the border line color
chart.Legend.FrameFormat.Border.LineColor = Color.Blue
'Sets the border line pattern
chart.Legend.FrameFormat.Border.LinePattern = OfficeChartLinePattern.DashDot
'Sets the legend border line weight
chart.Legend.FrameFormat.Border.LineWeight = OfficeChartLineWeight.Wide
'Sets the text area font weight
chart.Legend.TextArea.Bold = True
'Sets the legend text area fore color
chart.Legend.TextArea.Color = OfficeKnownColors.Bright_green
'Sets the legend text area font name
chart.Legend.TextArea.FontName = "Times New Roman"
'Sets the legend text area font size
chart.Legend.TextArea.Size = 20
'Sets the legend text area' strike through
chart.Legend.TextArea.Strikethrough = True
'Modifies the legend entry
chart.Legend.LegendEntries(0).IsDeleted = True
'Modifies the legend layout height
chart.Legend.Layout.Height = 200
'Modifies the legend layout height mode
chart.Legend.Layout.HeightMode = LayoutModes.factor
'Modifies the legend layout left position
chart.Legend.Layout.Left = 100
'Modifies the legend layout left mode
```



```

chart.Legend.Layout.LeftMode = LayoutModes.factor
'Modifies the legend layout top position
chart.Legend.Layout.Top = 100
'Modifies the legend layout top mode
chart.Legend.Layout.TopMode = LayoutModes.factor
'Modifies the legend layout width
chart.Legend.Layout.Width = 300
'Modifies the legend layout width mode
chart.Legend.Layout.WidthMode = LayoutModes.factor
'Saves the Presentation
pptxDoc.Save("ModifiedChart.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Opens a Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Gets the chart from the first slide
IPresentationChart chart = pptxDoc.Slides[0].Charts[0] as
IPresentationChart;
//Sets border settings
chart.PlotArea.Border.AutoFormat = false;
//Sets the auto line color
chart.PlotArea.Border.IsAutoLineColor = false;
//Sets the border line color
chart.PlotArea.Border.LineColor = Color.Blue;
//Sets the border line pattern
chart.PlotArea.Border.LinePattern = OfficeChartLinePattern.DashDot;
//Sets the border line weight
chart.PlotArea.Border.LineWeight = OfficeChartLineWeight.Wide;
//Sets the border transparency
chart.PlotArea.Border.Transparency = 0.6;
//Sets the plot area's fill type
chart.PlotArea.Fill.FillType = OfficeFillType.SolidColor;
//Sets the plot area's fill color
chart.PlotArea.Fill.ForeColor = Color.LightPink;
//Sets the plot area shadow presence
chart.PlotArea.Shadow.ShadowInnerPresets =
Office2007ChartPresetsInner.InsideDiagonalTopLeft;
//Sets the legend position
chart.Legend.Position = OfficeLegendPosition.Left;
//Sets the layout inclusion
chart.Legend.IncludeInLayout = true;
//Sets the legend border format
chart.Legend.FrameFormat.Border.AutoFormat = false;
//Sets the legend border auto line color
chart.Legend.FrameFormat.Border.IsAutoLineColor = false;
//Sets the border line color
chart.Legend.FrameFormat.Border.LineColor = Color.Blue;
//Sets the border line pattern
chart.Legend.FrameFormat.Border.LinePattern =
OfficeChartLinePattern.DashDot;
//Sets the legend border line weight
chart.Legend.FrameFormat.Border.LineWeight = OfficeChartLineWeight.Wide;
//Sets the font weight for legend text

```

```

chart.Legend.TextArea.Bold = true;
//Sets the fore color to legend text
chart.Legend.TextArea.Color = OfficeKnownColors.Bright_green;
//Sets the font name for legend text
chart.Legend.TextArea.FontName = "Times New Roman";
//Sets the font size for legend text
chart.Legend.TextArea.Size = 20;
//Sets the legend text area' strike through
chart.Legend.TextArea.Strikethrough = true;
//Modifies the legend entry
chart.Legend.LegendEntries[0].IsDeleted = true;
//Modifies the legend layout height
chart.Legend.Layout.Height = 200;
//Modifies the legend layout height mode
chart.Legend.Layout.HeightMode = LayoutModes.factor;
//Modifies the legend layout left position
chart.Legend.Layout.Left = 100;
//Modifies the legend layout left mode
chart.Legend.Layout.LeftMode = LayoutModes.factor;
//Modifies the legend layout top position
chart.Legend.Layout.Top = 100;
//Modifies the legend layout top mode
chart.Legend.Layout.TopMode = LayoutModes.factor;
//Modifies the legend layout width
chart.Legend.Layout.Width = 300;
//Modifies the legend layout width mode
chart.Legend.Layout.WidthMode = LayoutModes.factor;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ModifiedChart";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the chart from the first slide
IPresentationChart chart = pptxDoc.Slides[0].Charts[0] as
IPresentationChart;
//Sets border settings
chart.PlotArea.Border.AutoFormat = false;
//Sets the auto line color
chart.PlotArea.Border.IsAutoLineColor = false;
//Sets the border line color
chart.PlotArea.Border.LineColor = Color.Blue;
//Sets the border line pattern
chart.PlotArea.Border.LinePattern = OfficeChartLinePattern.DashDot;
//Sets the border line weight
chart.PlotArea.Border.LineWeight = OfficeChartLineWeight.Wide;

```

```
//Sets the border transparency
chart.PlotArea.Border.Transparency = 0.6;
//Sets the plot area's fill type
chart.PlotArea.Fill.FillType = OfficeFillType.SolidColor;
//Sets the plot area's fill color
chart.PlotArea.Fill.ForeColor = Color.LightPink;
//Sets the plot area shadow presence
chart.PlotArea.Shadow.ShadowInnerPresets =
Office2007ChartPresetsInner.InsideDiagonalTopLeft;
//Sets the legend position
chart.Legend.Position = OfficeLegendPosition.Left;
//Sets the layout inclusion
chart.Legend.IncludeInLayout = true;
//Sets the legend border format
chart.Legend.FrameFormat.Border.AutoFormat = false;
//Sets the legend border auto line color
chart.Legend.FrameFormat.Border.IsAutoLineColor = false;
//Sets the border line color
chart.Legend.FrameFormat.Border.LineColor = Color.Blue;
//Sets the border line pattern
chart.Legend.FrameFormat.Border.LinePattern =
OfficeChartLinePattern.DashDot;
//Sets the legend border line weight
chart.Legend.FrameFormat.Border.LineWeight = OfficeChartLineWeight.Wide;
//Sets the font weight for legend text
chart.Legend.TextArea.Bold = true;
//Sets the fore color to legend text
chart.Legend.TextArea.Color = OfficeKnownColors.Bright_green;
//Sets the font name for legend text
chart.Legend.TextArea.FontName = "Times New Roman";
//Sets the font size for legend text
chart.Legend.TextArea.Size = 20;
//Sets the legend text area' strike through
chart.Legend.TextArea.Strikethrough = true;
//Modifies the legend entry
chart.Legend.LegendEntries[0].IsDeleted = true;
//Modifies the legend layout height
chart.Legend.Layout.Height = 200;
//Modifies the legend layout height mode
chart.Legend.Layout.HeightMode = LayoutModes.factor;
//Modifies the legend layout left position
chart.Legend.Layout.Left = 100;
//Modifies the legend layout left mode
chart.Legend.Layout.LeftMode = LayoutModes.factor;
//Modifies the legend layout top position
chart.Legend.Layout.Top = 100;
//Modifies the legend layout top mode
chart.Legend.Layout.TopMode = LayoutModes.factor;
//Modifies the legend layout width
chart.Legend.Layout.Width = 300;
//Modifies the legend layout width mode
chart.Legend.Layout.WidthMode = LayoutModes.factor;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("ModifiedChart.pptx",
FileStream.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
```

```
pptxDoc.Close();
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the chart from the first slide
IPresentationChart chart = pptxDoc.Slides[0].Charts[0] as
IPresentationChart;
//Sets border settings
chart.PlotArea.Border.AutoFormat = false;
//Sets the auto line color
chart.PlotArea.Border.IsAutoLineColor = false;
//Sets the border line color
chart.PlotArea.Border.LineColor = Color.Blue;
//Sets the border line pattern
chart.PlotArea.Border.LinePattern = OfficeChartLinePattern.DashDot;
//Sets the border line weight
chart.PlotArea.Border.LineWeight = OfficeChartLineWeight.Wide;
//Sets the border transparency
chart.PlotArea.Border.Transparency = 0.6;
//Sets the plot area's fill type
chart.PlotArea.Fill.FillType = OfficeFillType.SolidColor;
//Sets the plot area's fill color
chart.PlotArea.Fill.ForeColor = Color.LightPink;
//Sets the plot area shadow presence
chart.PlotArea.Shadow.ShadowInnerPresets =
Office2007ChartPresetsInner.InsideDiagonalTopLeft;
//Sets the legend position
chart.Legend.Position = OfficeLegendPosition.Left;
//Sets the layout inclusion
chart.Legend.IncludeInLayout = true;
//Sets the legend border format
chart.Legend.FrameFormat.Border.AutoFormat = false;
//Sets the legend border auto line color
chart.Legend.FrameFormat.Border.IsAutoLineColor = false;
//Sets the border line color
chart.Legend.FrameFormat.Border.LineColor = Color.Blue;
//Sets the border line pattern
chart.Legend.FrameFormat.Border.LinePattern =
OfficeChartLinePattern.DashDot;
//Sets the legend border line weight
chart.Legend.FrameFormat.Border.LineWeight = OfficeChartLineWeight.Wide;
//Sets the font weight for legend text
chart.Legend.TextArea.Bold = true;
//Sets the fore color to legend text
chart.Legend.TextArea.Color = OfficeKnownColors.Bright_green;
//Sets the font name for legend text
chart.Legend.TextArea.FontName = "Times New Roman";
//Sets the font size for legend text
chart.Legend.TextArea.Size = 20;
```

```

//Sets the legend text area' strike through
chart.Legend.TextArea.Strikethrough = true;
//Modifies the legend entry
chart.Legend.LegendEntries[0].IsDeleted = true;
//Modifies the legend layout height
chart.Legend.Layout.Height = 200;
//Modifies the legend layout height mode
chart.Legend.Layout.HeightMode = LayoutModes.factor;
//Modifies the legend layout left position
chart.Legend.Layout.Left = 100;
//Modifies the legend layout left mode
chart.Legend.Layout.LeftMode = LayoutModes.factor;
//Modifies the legend layout top position
chart.Legend.Layout.Top = 100;
//Modifies the legend layout top mode
chart.Legend.Layout.TopMode = LayoutModes.factor;
//Modifies the legend layout width
chart.Legend.Layout.Width = 300;
//Modifies the legend layout width mode
chart.Legend.Layout.WidthMode = LayoutModes.factor;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("ModifiedChart
.pptx", "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("ModifiedChart.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Positioning Chart Elements

The following code examples illustrate how to position the different chart elements.

C#

```

IPresentation pptxDoc = Presentation.Create();
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds chart to slide
IPresentationChart chart = slide.Shapes.AddChart(100, 120, 500, 300);
//Sets the data range of chart
chart.DataRange = chart.ChartData[1, 2, 4, 3];
//Sets data to the chart- RowIndex, columnIndex and data
chart.ChartData.SetValue(1, 2, "2012");
chart.ChartData.SetValue(2, 2, 330);
chart.ChartData.SetValue(3, 2, 490);
chart.ChartData.SetValue(4, 2, 700);

```

```

chart.ChartType = OfficeChartType.Area;
//Edge: Specifies the width or Height to be interpreted as right or bottom
of the chart element.
//Factor: Specifies the width or Height to be interpreted as the width or
height of the chart element.
chart.PlotArea.Layout.LeftMode = LayoutModes.auto;
chart.PlotArea.Layout.TopMode = LayoutModes.factor;
//Value in points should not be negative value when LayoutMode is Edge
//It can be a negative value, when the LayoutMode is Factor.
chart.ChartTitleArea.Layout.Left = 10;
chart.ChartTitleArea.Layout.Top = 100;
//Manually positions chart plot area
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.outer;
chart.PlotArea.Layout.LeftMode = LayoutModes.edge;
chart.PlotArea.Layout.TopMode = LayoutModes.edge;
//Manually positions chart legend
chart.Legend.Layout.LeftMode = LayoutModes.factor;
chart.Legend.Layout.TopMode = LayoutModes.factor;
//Saves the Presentation
pptxDoc.Save("Output.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

Dim pptxDoc As IPresentation = Presentation.Create()
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds chart to slide
Dim chart As IPresentationChart = slide.Shapes.AddChart(100, 120, 500, 300)
'Sets the data range of chart
chart.DataRange = chart.ChartData(1, 2, 4, 3)
'Sets data to the chart- RowIndex, columnIndex, and data
chart.ChartData.SetValue(1, 2, "2012")
chart.ChartData.SetValue(2, 2, 330)
chart.ChartData.SetValue(3, 2, 490)
chart.ChartData.SetValue(4, 2, 700)
chart.ChartType = OfficeChartType.Area
'Edge: Specifies the width or Height to be interpreted as right or bottom of
the chart element.
'Factor: Specifies the width or Height to be interpreted as the width or
height of the chart element.
chart.PlotArea.Layout.LeftMode = LayoutModes.auto
chart.PlotArea.Layout.TopMode = LayoutModes.factor
'Value in points should not be negative value, when LayoutMode is Edge
'It can be negative value, when the LayoutMode is Factor.
chart.ChartTitleArea.Layout.Left = 10
chart.ChartTitleArea.Layout.Top = 100
'Manually positions chart plot area
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.outer
chart.PlotArea.Layout.LeftMode = LayoutModes.edge
chart.PlotArea.Layout.TopMode = LayoutModes.edge
'Manually positions chart legend
chart.Legend.Layout.LeftMode = LayoutModes.factor
chart.Legend.Layout.TopMode = LayoutModes.factor
'Saves the Presentation
pptxDoc.Save("Output.pptx")

```

```
'Closes the Presentation
pptxDoc.Close()
```

UWP

```
IPresentation pptxDoc = Presentation.Create();
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds chart to slide
IPresentationChart chart = slide.Shapes.AddChart(100, 120, 500, 300);
//Sets the data range of chart
chart.DataRange = chart.ChartData[1, 2, 4, 3];
//Sets data to the chart- RowIndex, columnIndex and data
chart.ChartData.SetValue(1, 2, "2012");
chart.ChartData.SetValue(2, 2, 330);
chart.ChartData.SetValue(3, 2, 490);
chart.ChartData.SetValue(4, 2, 700);
chart.ChartType = OfficeChartType.Area;
//Edge: Specifies the width or Height to be interpreted as right or bottom
of the chart element.
//Factor: Specifies the width or Height to be interpreted as the width or
height of the chart element.
chart.PlotArea.Layout.LeftMode = LayoutModes.auto;
chart.PlotArea.Layout.TopMode = LayoutModes.factor;
//Value in points should not be negative value when LayoutMode is Edge
//It can be a negative value, when the LayoutMode is Factor.
chart.ChartTitleArea.Layout.Left = 10;
chart.ChartTitleArea.Layout.Top = 100;
//Manually positions chart plot area
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.outer;
chart.PlotArea.Layout.LeftMode = LayoutModes.edge;
chart.PlotArea.Layout.TopMode = LayoutModes.edge;
//Manually positions chart legend
chart.Legend.Layout.LeftMode = LayoutModes.factor;
chart.Legend.Layout.TopMode = LayoutModes.factor;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ModifiedChart";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
IPresentation pptxDoc = Presentation.Create();
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds chart to slide
IPresentationChart chart = slide.Shapes.AddChart(100, 120, 500, 300);
//Sets the data range of chart
chart.DataRange = chart.ChartData[1, 2, 4, 3];
//Sets data to the chart- RowIndex, columnIndex and data
chart.ChartData.SetValue(1, 2, "2012");
chart.ChartData.SetValue(2, 2, 330);
```

```

chart.ChartData.SetValue(3, 2, 490);
chart.ChartData.SetValue(4, 2, 700);
chart.ChartType = OfficeChartType.Area;
//Edge: Specifies the width or Height to be interpreted as right or bottom
of the chart element.
//Factor: Specifies the width or Height to be interpreted as the width or
height of the chart element.
chart.PlotArea.Layout.LeftMode = LayoutModes.auto;
chart.PlotArea.Layout.TopMode = LayoutModes.factor;
//Value in points should not be negative value when LayoutMode is Edge
//It can be a negative value, when the LayoutMode is Factor.
chart.ChartTitleArea.Layout.Left = 10;
chart.ChartTitleArea.Layout.Top = 100;
//Manually positions chart plot area
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.outer;
chart.PlotArea.Layout.LeftMode = LayoutModes.edge;
chart.PlotArea.Layout.TopMode = LayoutModes.edge;
//Manually positions chart legend
chart.Legend.Layout.LeftMode = LayoutModes.factor;
chart.Legend.Layout.TopMode = LayoutModes.factor;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

IPresentation pptxDoc = Presentation.Create();
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds chart to slide
IPresentationChart chart = slide.Shapes.AddChart(100, 120, 500, 300);
//Sets the data range of chart
chart.DataRange = chart.ChartData[1, 2, 4, 3];
//Sets data to the chart- RowIndex, columnIndex and data
chart.ChartData.SetValue(1, 2, "2012");
chart.ChartData.SetValue(2, 2, 330);
chart.ChartData.SetValue(3, 2, 490);
chart.ChartData.SetValue(4, 2, 700);
chart.ChartType = OfficeChartType.Area;
//Edge: Specifies the width or Height to be interpreted as right or bottom
of the chart element.
//Factor: Specifies the width or Height to be interpreted as the width or
height of the chart element.
chart.PlotArea.Layout.LeftMode = LayoutModes.auto;
chart.PlotArea.Layout.TopMode = LayoutModes.factor;
//Value in points should not be negative value when LayoutMode is Edge
//It can be a negative value, when the LayoutMode is Factor.
chart.ChartTitleArea.Layout.Left = 10;
chart.ChartTitleArea.Layout.Top = 100;
//Manually positions chart plot area
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.outer;
chart.PlotArea.Layout.LeftMode = LayoutModes.edge;
chart.PlotArea.Layout.TopMode = LayoutModes.edge;
//Manually positions chart legend
chart.Legend.Layout.LeftMode = LayoutModes.factor;

```



```

chart.Legend.Layout.TopMode = LayoutModes.factor;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Applying 3D Formats

Essential Presentation allows you to modify side wall, back wall, and floor settings of a 3-D chart. The following code example explains how to apply these settings to a 3-D chart.

C#

```

//Opens the Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Changes the chart type to 3D
chart.ChartType = OfficeChartType.Bar_Clustered_3D;
//Sets the rotation
chart.Rotation = 80;
//Sets the shadow angle
chart.SideWall.Shadow.Angle = 60;
//Sets the back wall border weight
chart.BackWall.Border.LineWeight = OfficeChartLineWeight.Narrow;
//Saves the Presentation
pptxDoc.Save("output.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Opens the Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Gets the first slide
Dim slide As ISlide = pptxDoc.Slides(0)
'Gets the chart in slide
Dim chart As IPresentationChart = TryCast(slide.Shapes(0),
IPresentationChart)

```

```

'Changes the chart type to 3D
chart.ChartType = OfficeChartType.Bar_Clustered_3D
'Sets the rotation
chart.Rotation = 80
'Sets the shadow angle
chart.SideWall.Shadow.Angle = 60
'Sets the back wall border weight
chart.BackWall.Border.LineWeight = OfficeChartLineWeight.Narrow
'Saves the Presentation
pptxDoc.Save("output.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Changes the chart type to 3D
chart.ChartType = OfficeChartType.Bar_Clustered_3D;
//Sets the rotation
chart.Rotation = 80;
//Sets the shadow angle
chart.SideWall.Shadow.Angle = 60;
//Sets the back wall border weight
chart.BackWall.Border.LineWeight = OfficeChartLineWeight.Narrow;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Changes the chart type to 3D

```

```

chart.ChartType = OfficeChartType.Bar_Clustered_3D;
//Sets the rotation
chart.Rotation = 80;
//Sets the shadow angle
chart.SideWall.Shadow.Angle = 60;
//Sets the back wall border weight
chart.BackWall.Border.LineWeight = OfficeChartLineWeight.Narrow;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Changes the chart type to 3D
chart.ChartType = OfficeChartType.Bar_Clustered_3D;
//Sets the rotation
chart.Rotation = 80;
//Sets the shadow angle
chart.SideWall.Shadow.Angle = 60;
//Sets the back wall border weight
chart.BackWall.Border.LineWeight = OfficeChartLineWeight.Narrow;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Chart to Image conversion

The following code example demonstrates how to convert the charts in a Presentation slide to image.

Tips: You can specify the quality of the converted charts by setting the scaling mode. For more details on how to set the scaling mode, see [Converting PowerPoint presentation to Images](#)

C#

```
//Opens the Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Initializes the ChartToImageConverter class; this is mandatory
pptxDoc.ChartToImageConverter = new ChartToImageConverter();
//Sets the scaling mode for quality
pptxDoc.ChartToImageConverter.ScalingMode =
Syncfusion.OfficeChart.ScalingMode.Best;
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Creates a stream instance to store the image
MemoryStream stream = new MemoryStream();
//Saves the image to stream
chart.SaveAsImage(stream);
//Saves the stream to a file
using (FileStream fileStream = File.Create("ChartImage.png",
(int)stream.Length))
fileStream.Write(stream.ToArray(), 0, stream.ToArray().Length);
//Closes the stream
stream.Close();
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Opens the Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Initializes the ChartToImageConverter class; this is mandatory
pptxDoc.ChartToImageConverter = New ChartToImageConverter()
'Sets the scaling mode for quality
pptxDoc.ChartToImageConverter.ScalingMode =
Syncfusion.OfficeChart.ScalingMode.Best
'Gets the first slide
Dim slide As ISlide = pptxDoc.Slides(0)
'Gets the chart in slide
Dim chart As IPresentationChart = TryCast(slide.Shapes(0),
IPresentationChart)
'Creates a stream instance to store the image
Dim stream As New MemoryStream()
'Saves the image to stream
chart.SaveAsImage(stream)
'Saves the stream to a file
Using fileStream As FileStream = File.Create("ChartImage.png",
CInt(stream.Length))
fileStream.Write(stream.ToArray(), 0, stream.ToArray().Length)
End Using
'Closes the stream
```

```
stream.Close()  
'Closes the Presentation  
pptxDoc.Close()
```

Removing the chart from slide

The following code example demonstrates removing a chart from a slide.

C#

```
//Opens the Presentation  
IPresentation pptxDoc = Presentation.Open("Sample.pptx");  
//Gets the first slide  
ISlide slide = pptxDoc.Slides[0];  
//Gets the chart in slide  
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;  
//Removes the chart from slide  
slide.Shapes.Remove(chart as IShape);  
//Saves the Presentation  
pptxDoc.Save("output.pptx");  
//Closes the presentation  
pptxDoc.Close();
```

VB.NET

```
'Opens the Presentation  
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")  
'Gets the first slide  
Dim slide As ISlide = pptxDoc.Slides(0)  
'Gets the chart in slide  
Dim chart As IPresentationChart = TryCast(slide.Shapes(0),  
IPresentationChart)  
'Removes the chart from slide  
slide.Shapes.Remove(TryCast(chart, IShape))  
'Saves the Presentation  
pptxDoc.Save("output.pptx")  
'Closes the Presentation  
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker  
FileOpenPicker openPicker = new FileOpenPicker();  
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;  
openPicker.FileTypeFilter.Add(".pptx");  
//Creates a storage file from FileOpenPicker  
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();  
//Loads or open an PowerPoint Presentation  
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);  
//Gets the first slide  
ISlide slide = pptxDoc.Slides[0];  
//Gets the chart in slide  
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;  
//Removes the chart from slide  
slide.Shapes.Remove(chart as IShape);  
//Initializes FileSavePicker
```

```

FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Removes the chart from slide
slide.Shapes.Remove(chart as IShape);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the presentation
pptxDoc.Close();

```

XAMARIN

```

/"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide
ISlide slide = pptxDoc.Slides[0];
//Gets the chart in slide
IPresentationChart chart = slide.Shapes[0] as IPresentationChart;
//Removes the chart from slide
slide.Shapes.Remove(chart as IShape);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Creating a Scatter chart

The following code example demonstrates creating a Scatter chart.

C#

```
//Create an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds chart to the slide with position and size
IPresentationChart chart = slide.Charts.AddChart(100, 10, 700, 500);
//Set the chart type as Scatter_Markers
chart.ChartType = OfficeChartType.Scatter_Markers;
//Assign data
chart.DataRange = chart.ChartData[1, 1, 4, 2];
chart.IsSeriesInRows = false;
//Set data to the chart RowIndex, columnIndex, and data
chart.ChartData.SetValue(1, 1, "X-Axis");
chart.ChartData.SetValue(1, 2, "Y-Axis");
chart.ChartData.SetValue(2, 1, 1);
chart.ChartData.SetValue(3, 1, 5);
chart.ChartData.SetValue(4, 1, 10);
chart.ChartData.SetValue(2, 2, 10);
chart.ChartData.SetValue(3, 2, 5);
chart.ChartData.SetValue(4, 2, 1);
//Apply chart elements
//Set chart title
chart.ChartTitle = "Scatter Markers Chart";
//Set legend
chart.HasLegend = false;
//Set Datalabels
IOfficeChartSerie serie = chart.Series[0];
serie.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
serie.DataPoints.DefaultDataPoint.DataLabels.IsCategoryName = true;
//Saves the Presentation
pptxDoc.Save("output.pptx");
//Closes the presentation
pptxDoc.Close();
```

VB.NET

```
'Create an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create()
'Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds chart to the slide with position and size
IPresentationChart chart = slide.Charts.AddChart(100, 10, 700, 500)
'Set the chart type as Scatter Markers
```

```

chart.ChartType = OfficeChartType.Scatter_Markers
'Assign data
chart.DataRange = chart.ChartData[1, 1, 4, 2]
chart.IsSeriesInRows = false
'Set data to the chart RowIndex, columnIndex, and data
chart.ChartData.SetValue(1, 1, "X-Axis")
chart.ChartData.SetValue(1, 2, "Y-Axis")
chart.ChartData.SetValue(2, 1, 1)
chart.ChartData.SetValue(3, 1, 5)
chart.ChartData.SetValue(4, 1, 10)
chart.ChartData.SetValue(2, 2, 10)
chart.ChartData.SetValue(3, 2, 5)
chart.ChartData.SetValue(4, 2, 1)
'Apply chart elements
'Set chart title
chart.ChartTitle = "Scatter Markers Chart"
'Set legend
chart.HasLegend = false
'Set Datalabels
IOfficeChartSerie serie = chart.Series[0]
serie.DataPoints.DefaultDataPoint.DataLabels.IsValue = true
serie.DataPoints.DefaultDataPoint.DataLabels.IsCategoryName = true
'Saves the Presentation
pptxDoc.Save("output.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Create an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds chart to the slide with position and size
IPresentationChart chart = slide.Charts.AddChart(100, 10, 700, 500);
//Set the chart type as Scatter_Markers
chart.ChartType = OfficeChartType.Scatter_Markers;
//Assign data
chart.DataRange = chart.ChartData[1, 1, 4, 2];
chart.IsSeriesInRows = false;
//Set data to the chart RowIndex, columnIndex, and data
chart.ChartData.SetValue(1, 1, "X-Axis");
chart.ChartData.SetValue(1, 2, "Y-Axis");
chart.ChartData.SetValue(2, 1, 1);
chart.ChartData.SetValue(3, 1, 5);
chart.ChartData.SetValue(4, 1, 10);
chart.ChartData.SetValue(2, 2, 10);
chart.ChartData.SetValue(3, 2, 5);
chart.ChartData.SetValue(4, 2, 1);
//Apply chart elements
//Set chart title
chart.ChartTitle = "Scatter Markers Chart";
//Set legend
chart.HasLegend = false;
//Set Datalabels
IOfficeChartSerie serie = chart.Series[0];

```



```

serie.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
serie.DataPoints.DefaultDataPoint.DataLabels.IsCategoryName = true;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Create an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds chart to the slide with position and size
IPresentationChart chart = slide.Charts.AddChart(100, 10, 700, 500);
//Set the chart type as Scatter_Markers
chart.ChartType = OfficeChartType.Scatter_Markers;
//Assign data
chart.DataRange = chart.ChartData[1, 1, 4, 2];
chart.IsSeriesInRows = false;
//Set data to the chart RowIndex, columnIndex, and data
chart.ChartData.SetValue(1, 1, "X-Axis");
chart.ChartData.SetValue(1, 2, "Y-Axis");
chart.ChartData.SetValue(2, 1, 1);
chart.ChartData.SetValue(3, 1, 5);
chart.ChartData.SetValue(4, 1, 10);
chart.ChartData.SetValue(2, 2, 10);
chart.ChartData.SetValue(3, 2, 5);
chart.ChartData.SetValue(4, 2, 1);
//Apply chart elements
//Set chart title
chart.ChartTitle = "Scatter Markers Chart";
//Set legend
chart.HasLegend = false;
//Set Datalabels
IOfficeChartSerie serie = chart.Series[0];
serie.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
serie.DataPoints.DefaultDataPoint.DataLabels.IsCategoryName = true;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the presentation
pptxDoc.Close();

```

XAMARIN

```

//Create an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);

```

```

//Adds chart to the slide with position and size
IPresentationChart chart = slide.Charts.AddChart(100, 10, 700, 500);
//Set the chart type as Scatter_Markers
chart.ChartType = OfficeChartType.Scatter_Markers;
//Assign data
chart.DataRange = chart.ChartData[1, 1, 4, 2];
chart.IsSeriesInRows = false;
//Set data to the chart RowIndex, columnIndex, and data
chart.ChartData.SetValue(1, 1, "X-Axis");
chart.ChartData.SetValue(1, 2, "Y-Axis");
chart.ChartData.SetValue(2, 1, 1);
chart.ChartData.SetValue(3, 1, 5);
chart.ChartData.SetValue(4, 1, 10);
chart.ChartData.SetValue(2, 2, 10);
chart.ChartData.SetValue(3, 2, 5);
chart.ChartData.SetValue(4, 2, 1);
//Apply chart elements
//Set chart title
chart.ChartTitle = "Scatter Markers Chart";
//Set legend
chart.HasLegend = false;
//Set Datalabels
IOfficeChartSerie serie = chart.Series[0];
serie.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
serie.DataPoints.DefaultDataPoint.DataLabels.IsCategoryName = true;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

PowerPoint 2016 Charts

Essential Presentation supports creating and manipulating new and modern chart types such as waterfall, histogram, pareto, box and whisker, tree map, and sunburst, which are introduced in Microsoft PowerPoint 2016.

Funnel

[Funnel](#) charts show values across multiple stages in a process. Refer to the following code example to create a Funnel chart.

C#

```

using (IPresentation pptxDoc = Presentation.Create())
{
    ISlide slidel1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
    //Create a chart
    IPresentationChart chart = slidel1.Charts.AddChart(30, 50, 600, 300);
    //Set chart type as Funnel
    chart.ChartType = OfficeChartType.Funnel;
    //Set the chart title
    chart.ChartTitle = "Funnel";
    //Assign data
    chart.DataRange = chart.ChartData[1, 1, 6, 2];
    chart.IsSeriesInRows = false;
    //Set data
    chart.ChartData.SetValue(1, 1, "Web sales");
    chart.ChartData.SetValue(1, 2, "Users count");
    chart.ChartData.SetValue(2, 1, "Website Visits");
    chart.ChartData.SetValue(2, 2, "15600");
    chart.ChartData.SetValue(3, 1, "Downloads");
    chart.ChartData.SetValue(3, 2, "8000");
    chart.ChartData.SetValue(4, 1, "Requested price list");
    chart.ChartData.SetValue(4, 2, "6000");
    chart.ChartData.SetValue(5, 1, "Invoice sent");
    chart.ChartData.SetValue(5, 2, "2000");
    chart.ChartData.SetValue(6, 1, "Finalized");
    chart.ChartData.SetValue(6, 2, "1000");
    //Formatting the legend and data label option
    chart.HasLegend = false;
    IOfficeChartSerie serie = chart.Series[0];
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
    //Save and close the presentation
    pptxDoc.Save("FunnelChart.pptx");
    pptxDoc.Close();
}

```

VB.NET

```

'Creates a PowerPoint instance
Dim pptxDoc As IPresentation = Presentation.Create()
Dim slidel1 As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Create a chart
Dim chart As IPresentationChart = slidel1.Charts.AddChart(30, 50, 600, 300)
'Set chart type as Funnel
chart.ChartType = OfficeChartType.Funnel
'Set the chart title
chart.ChartTitle = "Funnel"
'Assign data
chart.DataRange = chart.ChartData(1, 1, 6, 2)
chart.IsSeriesInRows = False
'Set data
chart.ChartData.SetValue(1, 1, "Web sales")
chart.ChartData.SetValue(1, 2, "Users count")
chart.ChartData.SetValue(2, 1, "Website Visits")
chart.ChartData.SetValue(2, 2, "15600")
chart.ChartData.SetValue(3, 1, "Downloads")
chart.ChartData.SetValue(3, 2, "8000")

```

```

chart.ChartData.SetValue(4, 1, "Requested price list")
chart.ChartData.SetValue(4, 2, "6000")
chart.ChartData.SetValue(5, 1, "Invoice sent")
chart.ChartData.SetValue(5, 2, "2000")
chart.ChartData.SetValue(6, 1, "Finalized")
chart.ChartData.SetValue(6, 2, "1000")
'Formatting the legend and data label option
chart.HasLegend = False
Dim serie As IOOfficeChartSerie = chart.Series(0)
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.IsValue = True
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.Size = 8
'Save and close the presentation
pptxDoc.Save("FunnelChart.pptx")
pptxDoc.Close()

```

UWP

```

//Creates a Presentation instance
IPresentation pptxDoc = Presentation.Create();
//Adds a slide to Presentation
ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Create a chart
IPresentationChart chart = slide1.Charts.AddChart(30, 50, 600, 300);
//Set chart type as Funnel
chart.ChartType = OfficeChartType.Funnel;
//Set the chart title
chart.ChartTitle = "Funnel";
//Assign data
chart.DataRange = chart.ChartData[1, 1, 6, 2];
chart.IsSeriesInRows = false;
//Set data
chart.ChartData.SetValue(1, 1, "Web sales");
chart.ChartData.SetValue(1, 2, "Users count");
chart.ChartData.SetValue(2, 1, "Website Visits");
chart.ChartData.SetValue(2, 2, "15600");
chart.ChartData.SetValue(3, 1, "Downloads");
chart.ChartData.SetValue(3, 2, "8000");
chart.ChartData.SetValue(4, 1, "Requested price list");
chart.ChartData.SetValue(4, 2, "6000");
chart.ChartData.SetValue(5, 1, "Invoice sent");
chart.ChartData.SetValue(5, 2, "2000");
chart.ChartData.SetValue(6, 1, "Finalized");
chart.ChartData.SetValue(6, 2, "1000");
//Formatting the legend and data label option
chart.HasLegend = false;
IOOfficeChartSerie serie = chart.Series[0];
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "FunnelChart";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();

```

```
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

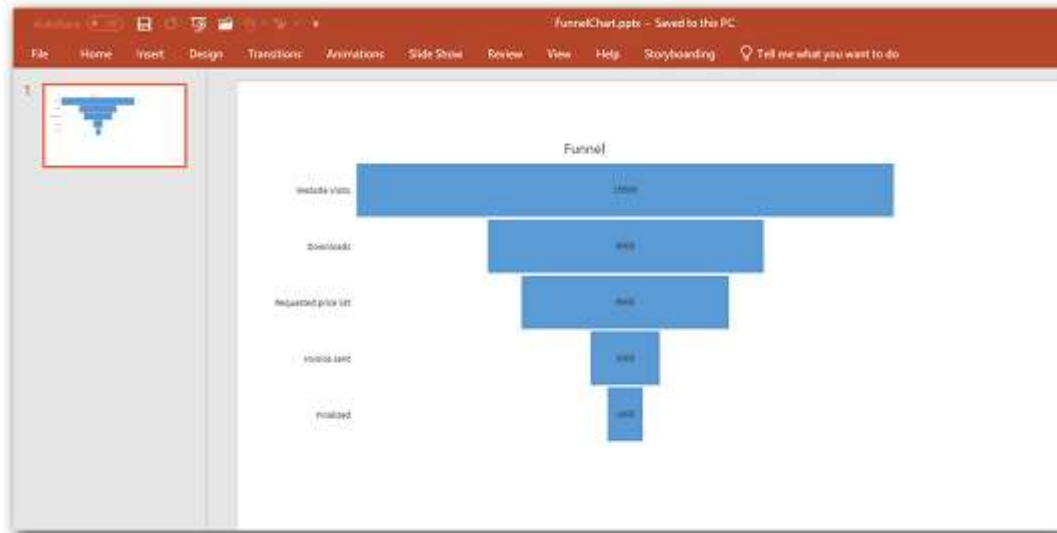
```
//Creates a new instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a slide to Presentation
ISlide slidel = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Create a chart
IPresentationChart chart = slidel.Charts.AddChart(30, 50, 600, 300);
//Set chart type as Funnel
chart.ChartType = OfficeChartType.Funnel;
//Set the chart title
chart.ChartTitle = "Funnel";
//Assign data
chart.DataRange = chart.ChartData[1, 1, 6, 2];
chart.IsSeriesInRows = false;
//Set data
chart.ChartData.SetValue(1, 1, "Web sales");
chart.ChartData.SetValue(1, 2, "Users count");
chart.ChartData.SetValue(2, 1, "Website Visits");
chart.ChartData.SetValue(2, 2, "15600");
chart.ChartData.SetValue(3, 1, "Downloads");
chart.ChartData.SetValue(3, 2, "8000");
chart.ChartData.SetValue(4, 1, "Requested price list");
chart.ChartData.SetValue(4, 2, "6000");
chart.ChartData.SetValue(5, 1, "Invoice sent");
chart.ChartData.SetValue(5, 2, "2000");
chart.ChartData.SetValue(6, 1, "Finalized");
chart.ChartData.SetValue(6, 2, "1000");
//Formatting the legend and data label option
chart.HasLegend = false;
IOfficeChartSerie serie = chart.Series[0];
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
//Save and close the presentation
pptxDoc.Save("FunnelChart.pptx");
pptxDoc.Close();
```

XAMARIN

```
//Creates a new instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a slide to Presentation
ISlide slidel = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Create a chart
IPresentationChart chart = slidel.Charts.AddChart(30, 50, 600, 300);
//Set chart type as Funnel
chart.ChartType = OfficeChartType.Funnel;
//Set the chart title
chart.ChartTitle = "Funnel";
//Assign data
chart.DataRange = chart.ChartData[1, 1, 6, 2];
chart.IsSeriesInRows = false;
//Set data
```

```
chart.ChartData.SetValue(1, 1, "Web sales");
chart.ChartData.SetValue(1, 2, "Users count");
chart.ChartData.SetValue(2, 1, "Website Visits");
chart.ChartData.SetValue(2, 2, "15600");
chart.ChartData.SetValue(3, 1, "Downloads");
chart.ChartData.SetValue(3, 2, "8000");
chart.ChartData.SetValue(4, 1, "Requested price list");
chart.ChartData.SetValue(4, 2, "6000");
chart.ChartData.SetValue(5, 1, "Invoice sent");
chart.ChartData.SetValue(5, 2, "2000");
chart.ChartData.SetValue(6, 1, "Finalized");
chart.ChartData.SetValue(6, 2, "1000");
//Formatting the legend and data label option
chart.HasLegend = false;
IOfficeChartSerie serie = chart.Series[0];
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("FunnelChart.p
ptx", "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("FunnelChart.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

The following screenshot shows the output of previous code.



Box and Whisker

[Box and Whisker](#) chart shows distribution of data into quartiles, highlighting the mean and outliers. Box and Whisker charts are most commonly used in statistical analysis. Refer to the following code example to create the Box and Whisker chart.

C#

```
private static void TestBox_Whisker()
{
    using (IPresentation pptxDoc = Presentation.Create())
    {
        ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
        IPresentationChart chart = slide1.Charts.AddChart(50, 50, 600, 400);
        chart.ChartTitle = "Test Scores";
        chart.ChartType = OfficeChartType.BoxAndWhisker;
        //Assign data
        chart.DataRange = chart.ChartData[1, 1, 16, 4];
        chart.IsSeriesInRows = false;
        //Set data to the chart RowIndex, columnIndex, and data
        SetChartData(chart);
        //Box and Whisker settings on first series
        IOfficeChartSeries seriesA = chart.Series[0];
        seriesA.SeriesFormat.ShowInnerPoints = false;
        seriesA.SeriesFormat.ShowOutlierPoints = true;
        seriesA.SeriesFormat.ShowMeanMarkers = true;
        seriesA.SeriesFormat.ShowMeanLine = false;
        seriesA.SeriesFormat.QuartileCalculationType =
            QuartileCalculation.ExclusiveMedian;
        //Box and Whisker settings on second series
        IOfficeChartSeries seriesB = chart.Series[1];
        seriesB.SeriesFormat.ShowInnerPoints = false;
        seriesB.SeriesFormat.ShowOutlierPoints = true;
        seriesB.SeriesFormat.ShowMeanMarkers = true;
        seriesB.SeriesFormat.ShowMeanLine = false;
        seriesB.SeriesFormat.QuartileCalculationType =
            QuartileCalculation.InclusiveMedian;
        //Box and Whisker settings on third series
        IOfficeChartSeries seriesC = chart.Series[2];
    }
}
```

```

seriesC.SerieFormat.ShowInnerPoints = false;
seriesC.SerieFormat.ShowOutlierPoints = true;
seriesC.SerieFormat.ShowMeanMarkers = true;
seriesC.SerieFormat.ShowMeanLine = false;
seriesC.SerieFormat.QuartileCalculationType =
QuartileCalculation.ExclusiveMedian;
//Save and close the presentation
pptxDoc.Save("BoxAndWhisker.pptx");
pptxDoc.Close();
}
}
/// <summary>
/// Set the values for the chart
/// </summary>
/// <param name="chart">Represent the instance of the Presentation
chart</param>
private static void SetChartData(IPresentationChart chart)
{
chart.ChartData.SetValue(1, 1, "Course");
chart.ChartData.SetValue(1, 2, "SchoolA");
chart.ChartData.SetValue(1, 3, "SchoolB");
chart.ChartData.SetValue(1, 4, "SchoolC");
chart.ChartData.SetValue(2, 1, "English");
chart.ChartData.SetValue(2, 2, 63);
chart.ChartData.SetValue(2, 3, 53);
chart.ChartData.SetValue(2, 4, 45);
chart.ChartData.SetValue(3, 1, "Physics");
chart.ChartData.SetValue(3, 2, 61);
chart.ChartData.SetValue(3, 3, 55);
chart.ChartData.SetValue(3, 4, 65);
chart.ChartData.SetValue(4, 1, "English");
chart.ChartData.SetValue(4, 2, 63);
chart.ChartData.SetValue(4, 3, 50);
chart.ChartData.SetValue(4, 4, 65);
chart.ChartData.SetValue(5, 1, "Math");
chart.ChartData.SetValue(5, 2, 62);
chart.ChartData.SetValue(5, 3, 51);
chart.ChartData.SetValue(5, 4, 64);
chart.ChartData.SetValue(6, 1, "English");
chart.ChartData.SetValue(6, 2, 46);
chart.ChartData.SetValue(6, 3, 53);
chart.ChartData.SetValue(6, 4, 66);
chart.ChartData.SetValue(7, 1, "English");
chart.ChartData.SetValue(7, 2, 58);
chart.ChartData.SetValue(7, 3, 56);
chart.ChartData.SetValue(7, 4, 67);
chart.ChartData.SetValue(8, 1, "Math");
chart.ChartData.SetValue(8, 2, 62);
chart.ChartData.SetValue(8, 3, 53);
chart.ChartData.SetValue(8, 4, 66);
chart.ChartData.SetValue(9, 1, "Math");
chart.ChartData.SetValue(9, 2, 62);
chart.ChartData.SetValue(9, 3, 53);
chart.ChartData.SetValue(9, 4, 66);
chart.ChartData.SetValue(10, 1, "English");
chart.ChartData.SetValue(10, 2, 63);
chart.ChartData.SetValue(10, 3, 54);

```



```

chart.ChartData.SetValue(10, 4, 64);
chart.ChartData.SetValue(11, 1, "English");
chart.ChartData.SetValue(11, 2, 63);
chart.ChartData.SetValue(11, 3, 52);
chart.ChartData.SetValue(11, 4, 67);
chart.ChartData.SetValue(12, 1, "Physics");
chart.ChartData.SetValue(12, 2, 60);
chart.ChartData.SetValue(12, 3, 56);
chart.ChartData.SetValue(12, 4, 64);
chart.ChartData.SetValue(13, 1, "English");
chart.ChartData.SetValue(13, 2, 60);
chart.ChartData.SetValue(13, 3, 56);
chart.ChartData.SetValue(13, 4, 64);
chart.ChartData.SetValue(14, 1, "Math");
chart.ChartData.SetValue(14, 2, 61);
chart.ChartData.SetValue(14, 3, 56);
chart.ChartData.SetValue(14, 4, 45);
chart.ChartData.SetValue(15, 1, "Math");
chart.ChartData.SetValue(15, 2, 63);
chart.ChartData.SetValue(15, 3, 58);
chart.ChartData.SetValue(15, 4, 64);
chart.ChartData.SetValue(16, 1, "English");
chart.ChartData.SetValue(16, 2, 59);
chart.ChartData.SetValue(16, 3, 54);
chart.ChartData.SetValue(16, 4, 65);
}

```

VB.NET

```

Sub TestBoxAndWhiskerChart()
    'Creates a PowerPoint instance
    Dim pptxDoc As IPresentation = Presentation.Create()
    Dim slide1 As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
    Dim chart As IPresentationChart = slide1.Charts.AddChart(50, 50, 600, 400)
    chart.ChartTitle = "Test Scores"
    chart.ChartType = OfficeChartType.BoxAndWhisker
    'Assign data
    chart.DataRange = chart.ChartData(1, 1, 16, 4)
    chart.IsSeriesInRows = False
    'Set data to the chart RowIndex, columnIndex, and data
    SetChartData(chart)
    'Box and Whisker settings on first series
    Dim seriesA As IOfficeChartSerie = chart.Series(0)
    seriesA.SerieFormat.ShowInnerPoints = False
    seriesA.SerieFormat.ShowOutlierPoints = True
    seriesA.SerieFormat.ShowMeanMarkers = True
    seriesA.SerieFormat.ShowMeanLine = False
    seriesA.SerieFormat.QuartileCalculationType =
    QuartileCalculation.ExclusiveMedian
    'Box and Whisker settings on second series
    Dim seriesB As IOfficeChartSerie = chart.Series(1)
    seriesB.SerieFormat.ShowInnerPoints = False
    seriesB.SerieFormat.ShowOutlierPoints = True
    seriesB.SerieFormat.ShowMeanMarkers = True
    seriesB.SerieFormat.ShowMeanLine = False

```

```

seriesB.SerieFormat.QuartileCalculationType =
QuartileCalculation.InclusiveMedian
'Box and Whisker settings on third series
Dim seriesC As IOOfficeChartSerie = chart.Series(2)
seriesC.SerieFormat.ShowInnerPoints = False
seriesC.SerieFormat.ShowOutlierPoints = True
seriesC.SerieFormat.ShowMeanMarkers = True
seriesC.SerieFormat.ShowMeanLine = False
seriesC.SerieFormat.QuartileCalculationType =
QuartileCalculation.ExclusiveMedian
'Save and close the presentation
pptxDoc.Save("BoxAndWhisker.pptx")
pptxDoc.Close()
End Sub
''' <summary>
''' Set the values for the chart
''' </summary>
''' <param name="chart">Represent the instance of the Presentation
chart</param>
Private Sub SetChartData(chart As IPresentationChart)
chart.ChartData.SetValue(1, 1, "Course")
chart.ChartData.SetValue(1, 2, "SchoolA")
chart.ChartData.SetValue(1, 3, "SchoolB")
chart.ChartData.SetValue(1, 4, "SchoolC")
chart.ChartData.SetValue(2, 1, "English")
chart.ChartData.SetValue(2, 2, 63)
chart.ChartData.SetValue(2, 3, 53)
chart.ChartData.SetValue(2, 4, 45)
chart.ChartData.SetValue(3, 1, "Physics")
chart.ChartData.SetValue(3, 2, 61)
chart.ChartData.SetValue(3, 3, 55)
chart.ChartData.SetValue(3, 4, 65)
chart.ChartData.SetValue(4, 1, "English")
chart.ChartData.SetValue(4, 2, 63)
chart.ChartData.SetValue(4, 3, 50)
chart.ChartData.SetValue(4, 4, 65)
chart.ChartData.SetValue(5, 1, "Math")
chart.ChartData.SetValue(5, 2, 62)
chart.ChartData.SetValue(5, 3, 51)
chart.ChartData.SetValue(5, 4, 64)
chart.ChartData.SetValue(6, 1, "English")
chart.ChartData.SetValue(6, 2, 46)
chart.ChartData.SetValue(6, 3, 53)
chart.ChartData.SetValue(6, 4, 66)
chart.ChartData.SetValue(7, 1, "English")
chart.ChartData.SetValue(7, 2, 58)
chart.ChartData.SetValue(7, 3, 56)
chart.ChartData.SetValue(7, 4, 67)
chart.ChartData.SetValue(8, 1, "Math")
chart.ChartData.SetValue(8, 2, 62)
chart.ChartData.SetValue(8, 3, 53)
chart.ChartData.SetValue(8, 4, 66)
chart.ChartData.SetValue(9, 1, "Math")
chart.ChartData.SetValue(9, 2, 62)
chart.ChartData.SetValue(9, 3, 53)
chart.ChartData.SetValue(9, 4, 66)
chart.ChartData.SetValue(10, 1, "English")

```

```

chart.ChartData.SetValue(10, 2, 63)
chart.ChartData.SetValue(10, 3, 54)
chart.ChartData.SetValue(10, 4, 64)
chart.ChartData.SetValue(11, 1, "English")
chart.ChartData.SetValue(11, 2, 63)
chart.ChartData.SetValue(11, 3, 52)
chart.ChartData.SetValue(11, 4, 67)
chart.ChartData.SetValue(12, 1, "Physics")
chart.ChartData.SetValue(12, 2, 60)
chart.ChartData.SetValue(12, 3, 56)
chart.ChartData.SetValue(12, 4, 64)
chart.ChartData.SetValue(13, 1, "English")
chart.ChartData.SetValue(13, 2, 60)
chart.ChartData.SetValue(13, 3, 56)
chart.ChartData.SetValue(13, 4, 64)
chart.ChartData.SetValue(14, 1, "Math")
chart.ChartData.SetValue(14, 2, 61)
chart.ChartData.SetValue(14, 3, 56)
chart.ChartData.SetValue(14, 4, 45)
chart.ChartData.SetValue(15, 1, "Math")
chart.ChartData.SetValue(15, 2, 63)
chart.ChartData.SetValue(15, 3, 58)
chart.ChartData.SetValue(15, 4, 64)
chart.ChartData.SetValue(16, 1, "English")
chart.ChartData.SetValue(16, 2, 59)
chart.ChartData.SetValue(16, 3, 54)
chart.ChartData.SetValue(16, 4, 65)
End Sub

```

UWP

```

private static void TestBox_Whisker()
{
    //Creates a new instance of PowerPoint Presentation
    IPresentation pptxDoc = Presentation.Create();
    //Adds a slide to Presentation
    ISlide slidel1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
    IPresentationChart chart = slidel1.Charts.AddChart(50, 50, 600, 400);
    chart.ChartTitle = "Test Scores";
    chart.ChartType = OfficeChartType.BoxAndWhisker;
    //Assign data
    chart.DataRange = chart.ChartData[1,1,16,4];
    chart.IsSeriesInRows = false;
    //Set data to the chart RowIndex, columnIndex, and data
    SetChartData(chart);
    //Box and Whisker settings on first series
    IOfficeChartSerie seriesA = chart.Series[0];
    seriesA.SerieFormat.ShowInnerPoints = false;
    seriesA.SerieFormat.ShowOutlierPoints = true;
    seriesA.SerieFormat.ShowMeanMarkers = true;
    seriesA.SerieFormat.ShowMeanLine = false;
    seriesA.SerieFormat.QuartileCalculationType =
    QuartileCalculation.ExclusiveMedian;
    //Box and Whisker settings on second series
    IOfficeChartSerie seriesB = chart.Series[1];
    seriesB.SerieFormat.ShowInnerPoints = false;
}

```

```

seriesB.SerieFormat.ShowOutlierPoints = true;
seriesB.SerieFormat.ShowMeanMarkers = true;
seriesB.SerieFormat.ShowMeanLine = false;
seriesB.SerieFormat.QuartileCalculationType =
QuartileCalculation.InclusiveMedian;
//Box and Whisker settings on third series
IOfficeChartSerie seriesC = chart.Series[2];
seriesC.SerieFormat.ShowInnerPoints = false;
seriesC.SerieFormat.ShowOutlierPoints = true;
seriesC.SerieFormat.ShowMeanMarkers = true;
seriesC.SerieFormat.ShowMeanLine = false;
seriesC.SerieFormat.QuartileCalculationType =
QuartileCalculation.ExclusiveMedian;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "BoxAndWhisker";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
}
/// <summary>
/// Set the values for the chart
/// </summary>
/// <param name="chart">Represent the instance of the Presentation
chart</param>
private static void SetChartData(IPresentationChart chart)
{
chart.ChartData.SetValue(1, 1, "Course");
chart.ChartData.SetValue(1, 2, "SchoolA");
chart.ChartData.SetValue(1, 3, "SchoolB");
chart.ChartData.SetValue(1, 4, "SchoolC");
chart.ChartData.SetValue(2, 1, "English");
chart.ChartData.SetValue(2, 2, 63);
chart.ChartData.SetValue(2, 3, 53);
chart.ChartData.SetValue(2, 4, 45);
chart.ChartData.SetValue(3, 1, "Physics");
chart.ChartData.SetValue(3, 2, 61);
chart.ChartData.SetValue(3, 3, 55);
chart.ChartData.SetValue(3, 4, 65);
chart.ChartData.SetValue(4, 1, "English");
chart.ChartData.SetValue(4, 2, 63);
chart.ChartData.SetValue(4, 3, 50);
chart.ChartData.SetValue(4, 4, 65);
chart.ChartData.SetValue(5, 1, "Math");
chart.ChartData.SetValue(5, 2, 62);
chart.ChartData.SetValue(5, 3, 51);
chart.ChartData.SetValue(5, 4, 64);
chart.ChartData.SetValue(6, 1, "English");
chart.ChartData.SetValue(6, 2, 46);
chart.ChartData.SetValue(6, 3, 53);
chart.ChartData.SetValue(6, 4, 66);
chart.ChartData.SetValue(7, 1, "English");
chart.ChartData.SetValue(7, 2, 58);

```

```

chart.ChartData.SetValue(7, 3, 56);
chart.ChartData.SetValue(7, 4, 67);
chart.ChartData.SetValue(8, 1, "Math");
chart.ChartData.SetValue(8, 2, 62);
chart.ChartData.SetValue(8, 3, 53);
chart.ChartData.SetValue(8, 4, 66);
chart.ChartData.SetValue(9, 1, "Math");
chart.ChartData.SetValue(9, 2, 62);
chart.ChartData.SetValue(9, 3, 53);
chart.ChartData.SetValue(9, 4, 66);
chart.ChartData.SetValue(10, 1, "English");
chart.ChartData.SetValue(10, 2, 63);
chart.ChartData.SetValue(10, 3, 54);
chart.ChartData.SetValue(10, 4, 64);
chart.ChartData.SetValue(11, 1, "English");
chart.ChartData.SetValue(11, 2, 63);
chart.ChartData.SetValue(11, 3, 52);
chart.ChartData.SetValue(11, 4, 67);
chart.ChartData.SetValue(12, 1, "Physics");
chart.ChartData.SetValue(12, 2, 60);
chart.ChartData.SetValue(12, 3, 56);
chart.ChartData.SetValue(12, 4, 64);
chart.ChartData.SetValue(13, 1, "English");
chart.ChartData.SetValue(13, 2, 60);
chart.ChartData.SetValue(13, 3, 56);
chart.ChartData.SetValue(13, 4, 64);
chart.ChartData.SetValue(14, 1, "Math");
chart.ChartData.SetValue(14, 2, 61);
chart.ChartData.SetValue(14, 3, 56);
chart.ChartData.SetValue(14, 4, 45);
chart.ChartData.SetValue(15, 1, "Math");
chart.ChartData.SetValue(15, 2, 63);
chart.ChartData.SetValue(15, 3, 58);
chart.ChartData.SetValue(15, 4, 64);
chart.ChartData.SetValue(16, 1, "English");
chart.ChartData.SetValue(16, 2, 59);
chart.ChartData.SetValue(16, 3, 54);
chart.ChartData.SetValue(16, 4, 65);
}

```

ASP.NET CORE

```

private static void TestBox_Whisker()
{
    using (IPresentation pptxDoc = Presentation.Create())
    {
        ISlide slidel1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
        IPresentationChart chart = slidel1.Charts.AddChart(50, 50, 600, 400);
        chart.ChartTitle = "Test Scores";
        chart.ChartType = OfficeChartType.BoxAndWhisker;
        //Assign data
        chart.DataRange = chart.ChartData[1, 1, 16, 4];
        chart.IsSeriesInRows = false;
        //Set data to the chart RowIndex, columnIndex, and data
        SetChartData(chart);
        //Box and Whisker settings on first series
    }
}

```

```

IOfficeChartSerie seriesA = chart.Series[0];
seriesA.SerieFormat.ShowInnerPoints = false;
seriesA.SerieFormat.ShowOutlierPoints = true;
seriesA.SerieFormat.ShowMeanMarkers = true;
seriesA.SerieFormat.ShowMeanLine = false;
seriesA.SerieFormat.QuartileCalculationType =
QuartileCalculation.ExclusiveMedian;
//Box and Whisker settings on second series
IOfficeChartSerie seriesB = chart.Series[1];
seriesB.SerieFormat.ShowInnerPoints = false;
seriesB.SerieFormat.ShowOutlierPoints = true;
seriesB.SerieFormat.ShowMeanMarkers = true;
seriesB.SerieFormat.ShowMeanLine = false;
seriesB.SerieFormat.QuartileCalculationType =
QuartileCalculation.InclusiveMedian;
//Box and Whisker settings on third series
IOfficeChartSerie seriesC = chart.Series[2];
seriesC.SerieFormat.ShowInnerPoints = false;
seriesC.SerieFormat.ShowOutlierPoints = true;
seriesC.SerieFormat.ShowMeanMarkers = true;
seriesC.SerieFormat.ShowMeanLine = false;
seriesC.SerieFormat.QuartileCalculationType =
QuartileCalculation.ExclusiveMedian;
//Save and close the presentation
pptxDoc.Save("BoxAndWhisker.pptx");
pptxDoc.Close();
}
}
/// <summary>
/// Set the values for the chart
/// </summary>
/// <param name="chart">Represent the instance of the Presentation
chart</param>
private static void SetChartData(IPresentationChart chart)
{
chart.ChartData.SetValue(1, 1, "Course");
chart.ChartData.SetValue(1, 2, "SchoolA");
chart.ChartData.SetValue(1, 3, "SchoolB");
chart.ChartData.SetValue(1, 4, "SchoolC");
chart.ChartData.SetValue(2, 1, "English");
chart.ChartData.SetValue(2, 2, 63);
chart.ChartData.SetValue(2, 3, 53);
chart.ChartData.SetValue(2, 4, 45);
chart.ChartData.SetValue(3, 1, "Physics");
chart.ChartData.SetValue(3, 2, 61);
chart.ChartData.SetValue(3, 3, 55);
chart.ChartData.SetValue(3, 4, 65);
chart.ChartData.SetValue(4, 1, "English");
chart.ChartData.SetValue(4, 2, 63);
chart.ChartData.SetValue(4, 3, 50);
chart.ChartData.SetValue(4, 4, 65);
chart.ChartData.SetValue(5, 1, "Math");
chart.ChartData.SetValue(5, 2, 62);
chart.ChartData.SetValue(5, 3, 51);
chart.ChartData.SetValue(5, 4, 64);
chart.ChartData.SetValue(6, 1, "English");
chart.ChartData.SetValue(6, 2, 46);

```

```
chart.ChartData.SetValue(6, 3, 53);
chart.ChartData.SetValue(6, 4, 66);
chart.ChartData.SetValue(7, 1, "English");
chart.ChartData.SetValue(7, 2, 58);
chart.ChartData.SetValue(7, 3, 56);
chart.ChartData.SetValue(7, 4, 67);
chart.ChartData.SetValue(8, 1, "Math");
chart.ChartData.SetValue(8, 2, 62);
chart.ChartData.SetValue(8, 3, 53);
chart.ChartData.SetValue(8, 4, 66);
chart.ChartData.SetValue(9, 1, "Math");
chart.ChartData.SetValue(9, 2, 62);
chart.ChartData.SetValue(9, 3, 53);
chart.ChartData.SetValue(9, 4, 66);
chart.ChartData.SetValue(10, 1, "English");
chart.ChartData.SetValue(10, 2, 63);
chart.ChartData.SetValue(10, 3, 54);
chart.ChartData.SetValue(10, 4, 64);
chart.ChartData.SetValue(11, 1, "English");
chart.ChartData.SetValue(11, 2, 63);
chart.ChartData.SetValue(11, 3, 52);
chart.ChartData.SetValue(11, 4, 67);
chart.ChartData.SetValue(12, 1, "Physics");
chart.ChartData.SetValue(12, 2, 60);
chart.ChartData.SetValue(12, 3, 56);
chart.ChartData.SetValue(12, 4, 64);
chart.ChartData.SetValue(13, 1, "English");
chart.ChartData.SetValue(13, 2, 60);
chart.ChartData.SetValue(13, 3, 56);
chart.ChartData.SetValue(13, 4, 64);
chart.ChartData.SetValue(14, 1, "Math");
chart.ChartData.SetValue(14, 2, 61);
chart.ChartData.SetValue(14, 3, 56);
chart.ChartData.SetValue(14, 4, 45);
chart.ChartData.SetValue(15, 1, "Math");
chart.ChartData.SetValue(15, 2, 63);
chart.ChartData.SetValue(15, 3, 58);
chart.ChartData.SetValue(15, 4, 64);
chart.ChartData.SetValue(16, 1, "English");
chart.ChartData.SetValue(16, 2, 59);
chart.ChartData.SetValue(16, 3, 54);
chart.ChartData.SetValue(16, 4, 65);
}
```

XAMARIN

```
private static void TestBox_Whisker()
{
    using(IPresentation pptxDoc = Presentation.Create())
    {
        ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
        IPresentationChart chart = slide1.Charts.AddChart(50, 50, 600, 400);
        chart.ChartTitle = "Test Scores";
        chart.ChartType = OfficeChartType.BoxAndWhisker;
        //Assign data
        chart.DataRange = chart.ChartData[1, 1, 16, 4];
    }
}
```

```

chart.IsSeriesInRows = false;
//Set data to the chart RowIndex, columnIndex, and data
SetChartData(chart);
//Box and Whisker settings on first series
IOfficeChartSerie seriesA = chart.Series[0];
seriesA.SerieFormat.ShowInnerPoints = false;
seriesA.SerieFormat.ShowOutlierPoints = true;
seriesA.SerieFormat.ShowMeanMarkers = true;
seriesA.SerieFormat.ShowMeanLine = false;
seriesA.SerieFormat.QuartileCalculationType =
QuartileCalculation.ExclusiveMedian;
//Box and Whisker settings on second series
IOfficeChartSerie seriesB = chart.Series[1];
seriesB.SerieFormat.ShowInnerPoints = false;
seriesB.SerieFormat.ShowOutlierPoints = true;
seriesB.SerieFormat.ShowMeanMarkers = true;
seriesB.SerieFormat.ShowMeanLine = false;
seriesB.SerieFormat.QuartileCalculationType =
QuartileCalculation.InclusiveMedian;
//Box and Whisker settings on third series
IOfficeChartSerie seriesC = chart.Series[2];
seriesC.SerieFormat.ShowInnerPoints = false;
seriesC.SerieFormat.ShowOutlierPoints = true;
seriesC.SerieFormat.ShowMeanMarkers = true;
seriesC.SerieFormat.ShowMeanLine = false;
seriesC.SerieFormat.QuartileCalculationType =
QuartileCalculation.ExclusiveMedian;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("BoxAndWhisker
.pptx.pptx", "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("BoxAndWhisker.pptx.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
}
}
/// <summary>
/// Set the values for the chart
/// </summary>
/// <param name="chart">Represent the instance of the Presentation
chart</param>
private static void SetChartData(IPresentationChart chart)
{
chart.ChartData.SetValue(1, 1, "Course");
chart.ChartData.SetValue(1, 2, "SchoolA");

```



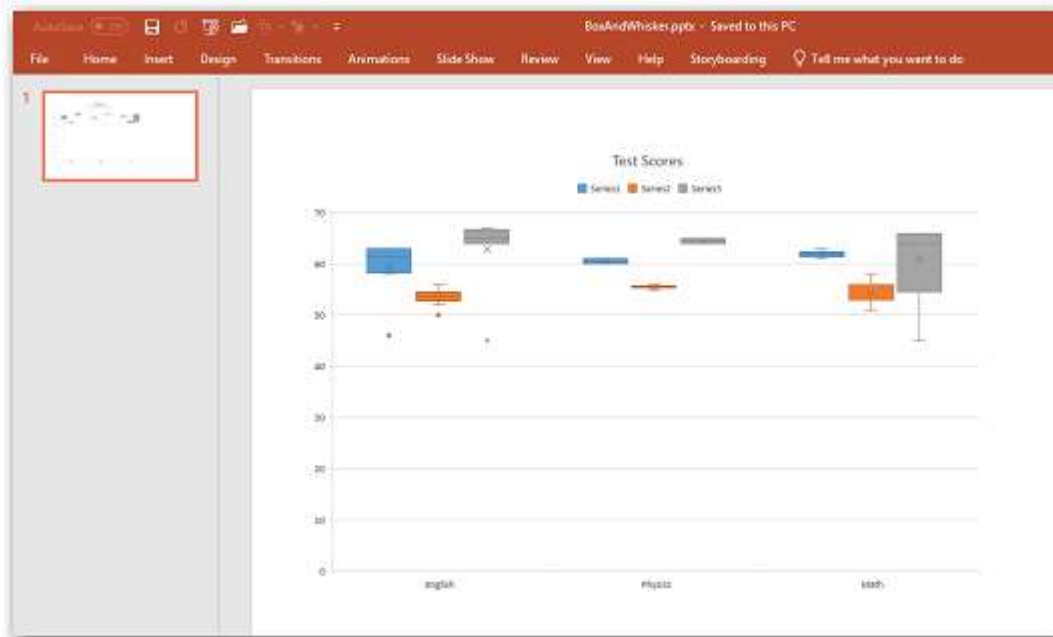
```
chart.ChartData.SetValue(1, 3, "SchoolB");
chart.ChartData.SetValue(1, 4, "SchoolC");
chart.ChartData.SetValue(2, 1, "English");
chart.ChartData.SetValue(2, 2, 63);
chart.ChartData.SetValue(2, 3, 53);
chart.ChartData.SetValue(2, 4, 45);
chart.ChartData.SetValue(3, 1, "Physics");
chart.ChartData.SetValue(3, 2, 61);
chart.ChartData.SetValue(3, 3, 55);
chart.ChartData.SetValue(3, 4, 65);
chart.ChartData.SetValue(4, 1, "English");
chart.ChartData.SetValue(4, 2, 63);
chart.ChartData.SetValue(4, 3, 50);
chart.ChartData.SetValue(4, 4, 65);
chart.ChartData.SetValue(5, 1, "Math");
chart.ChartData.SetValue(5, 2, 62);
chart.ChartData.SetValue(5, 3, 51);
chart.ChartData.SetValue(5, 4, 64);
chart.ChartData.SetValue(6, 1, "English");
chart.ChartData.SetValue(6, 2, 46);
chart.ChartData.SetValue(6, 3, 53);
chart.ChartData.SetValue(6, 4, 66);
chart.ChartData.SetValue(7, 1, "English");
chart.ChartData.SetValue(7, 2, 58);
chart.ChartData.SetValue(7, 3, 56);
chart.ChartData.SetValue(7, 4, 67);
chart.ChartData.SetValue(8, 1, "Math");
chart.ChartData.SetValue(8, 2, 62);
chart.ChartData.SetValue(8, 3, 53);
chart.ChartData.SetValue(8, 4, 66);
chart.ChartData.SetValue(9, 1, "Math");
chart.ChartData.SetValue(9, 2, 62);
chart.ChartData.SetValue(9, 3, 53);
chart.ChartData.SetValue(9, 4, 66);
chart.ChartData.SetValue(10, 1, "English");
chart.ChartData.SetValue(10, 2, 63);
chart.ChartData.SetValue(10, 3, 54);
chart.ChartData.SetValue(10, 4, 64);
chart.ChartData.SetValue(11, 1, "English");
chart.ChartData.SetValue(11, 2, 63);
chart.ChartData.SetValue(11, 3, 52);
chart.ChartData.SetValue(11, 4, 67);
chart.ChartData.SetValue(12, 1, "Physics");
chart.ChartData.SetValue(12, 2, 60);
chart.ChartData.SetValue(12, 3, 56);
chart.ChartData.SetValue(12, 4, 64);
chart.ChartData.SetValue(13, 1, "English");
chart.ChartData.SetValue(13, 2, 60);
chart.ChartData.SetValue(13, 3, 56);
chart.ChartData.SetValue(13, 4, 64);
chart.ChartData.SetValue(14, 1, "Math");
chart.ChartData.SetValue(14, 2, 61);
chart.ChartData.SetValue(14, 3, 56);
chart.ChartData.SetValue(14, 4, 45);
chart.ChartData.SetValue(15, 1, "Math");
chart.ChartData.SetValue(15, 2, 63);
chart.ChartData.SetValue(15, 3, 58);
```

```

chart.ChartData.SetValue(15, 4, 64);
chart.ChartData.SetValue(16, 1, "English");
chart.ChartData.SetValue(16, 2, 59);
chart.ChartData.SetValue(16, 3, 54);
chart.ChartData.SetValue(16, 4, 65);
}

```

The following screenshot shows the output of previous code.



Waterfall

Waterfall chart helps understand the finances of business owners by viewing profit and loss statements. You can quickly illustrate the line items in your financial data and get a clear picture of how each item is impacting your bottom line using a Waterfall chart. Refer to the following code to create a Waterfall chart.

C#

```

using (IPresentation pptxDoc = Presentation.Create())
{
    ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
    //Create a chart
    IPresentationChart chart = slide1.Charts.AddChart(50, 50, 700, 400);
    //Set chart type as Waterfall
    chart.ChartType = OfficeChartType.WaterFall;
    //Set data range
    chart.DataRange = chart.ChartData[1, 1, 8, 2];
    chart.IsSeriesInRows = false;
    chart.ChartData.SetValue(2, 1, "Start");
    chart.ChartData.SetValue(2, 2, 120000);
    chart.ChartData.SetValue(3, 1, "Product Revenue");
    chart.ChartData.SetValue(3, 2, 570000);
    chart.ChartData.SetValue(4, 1, "Service Revenue");
    chart.ChartData.SetValue(4, 2, 230000);
    chart.ChartData.SetValue(5, 1, "Positive Balance");
}

```

```

chart.ChartData.SetValue(5, 2, 920000);
chart.ChartData.SetValue(6, 1, "Fixed Costs");
chart.ChartData.SetValue(6, 2, -345000);
chart.ChartData.SetValue(7, 1, "Variable Costs");
chart.ChartData.SetValue(7, 2, -230000);
chart.ChartData.SetValue(8, 1, "Total");
chart.ChartData.SetValue(8, 2, 345000);
//Data point settings as total in chart
IOfficeChartSerie series = chart.Series[0];
chart.Series[0].DataPoints[2].SetAsTotal = true;
chart.Series[0].DataPoints[5].SetAsTotal = true;
//Showing the connector lines between data points
chart.Series[0].SerieFormat.ShowConnectorLines = true;
//Set the chart title
chart.ChartTitle = "Company Profit (in USD)";
//Formatting data label and legend option
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
chart.Legend.Position = OfficeLegendPosition.Right;
//Save and close the presentation
pptxDoc.Save("WaterFall.pptx");
pptxDoc.Close();
}

```

VB.NET

```

'Creates a PowerPoint instance
Dim pptxDoc As IPresentation = Presentation.Create()
Dim slidel As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Create a chart
Dim chart As IPresentationChart = slidel.Charts.AddChart(50, 50, 700, 400)
'Set chart type as Waterfall
chart.ChartType = OfficeChartType.WaterFall
'Set data range
chart.DataRange = chart.ChartData(1, 1, 8, 2)
chart.IsSeriesInRows = False
chart.ChartData.SetValue(2, 1, "Start")
chart.ChartData.SetValue(2, 2, 120000)
chart.ChartData.SetValue(3, 1, "Product Revenue")
chart.ChartData.SetValue(3, 2, 570000)
chart.ChartData.SetValue(4, 1, "Service Revenue")
chart.ChartData.SetValue(4, 2, 230000)
chart.ChartData.SetValue(5, 1, "Positive Balance")
chart.ChartData.SetValue(5, 2, 920000)
chart.ChartData.SetValue(6, 1, "Fixed Costs")
chart.ChartData.SetValue(6, 2, -345000)
chart.ChartData.SetValue(7, 1, "Variable Costs")
chart.ChartData.SetValue(7, 2, -230000)
chart.ChartData.SetValue(8, 1, "Total")
chart.ChartData.SetValue(8, 2, 345000)
'Data point settings as total in chart
Dim series As IOfficeChartSerie = chart.Series(0)
chart.Series(0).DataPoints(2).SetAsTotal = True
chart.Series(0).DataPoints(5).SetAsTotal = True
'Showing the connector lines between data points
chart.Series(0).SerieFormat.ShowConnectorLines = True

```

```

'Set the chart title
chart.ChartTitle = "Company Profit (in USD)"
'Formatting data label and legend option
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.IsValue = True
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.Size = 8
chart.Legend.Position = OfficeLegendPosition.Right
'Save and close the presentation
pptxDoc.Save("WaterFall.pptx")
pptxDoc.Close()

```

UWP

```

//Creates a new instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
ISlide slidel = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Create a chart
IPresentationChart chart = slidel.Charts.AddChart(50, 50, 700, 400);
//Set chart type as Waterfall
chart.ChartType = OfficeChartType.WaterFall;
//Set data range
chart.DataRange = chart.ChartData[1, 1, 8, 2];
chart.IsSeriesInRows = false;
chart.ChartData.SetValue(2, 1, "Start");
chart.ChartData.SetValue(2, 2, 120000);
chart.ChartData.SetValue(3, 1, "Product Revenue");
chart.ChartData.SetValue(3, 2, 570000);
chart.ChartData.SetValue(4, 1, "Service Revenue");
chart.ChartData.SetValue(4, 2, 230000);
chart.ChartData.SetValue(5, 1, "Positive Balance");
chart.ChartData.SetValue(5, 2, 920000);
chart.ChartData.SetValue(6, 1, "Fixed Costs");
chart.ChartData.SetValue(6, 2, -345000);
chart.ChartData.SetValue(7, 1, "Variable Costs");
chart.ChartData.SetValue(7, 2, -230000);
chart.ChartData.SetValue(8, 1, "Total");
chart.ChartData.SetValue(8, 2, 345000);
//Data point settings as total in chart
IOfficeChartSerie series = chart.Series[0];
chart.Series[0].DataPoints[2].SetAsTotal = true;
chart.Series[0].DataPoints[5].SetAsTotal = true;
//Showing the connector lines between data points
chart.Series[0].SerieFormat.ShowConnectorLines = true;
//Set the chart title
chart.ChartTitle = "Company Profit (in USD)";
//Formatting data label and legend option
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
chart.Legend.Position = OfficeLegendPosition.Right;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "WaterFall";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();

```

```
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Creates a new instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add slide to the presentation
ISlide slidel = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Create a chart
IPresentationChart chart = slidel.Charts.AddChart(50, 50, 700, 400);
//Set chart type as Waterfall
chart.ChartType = OfficeChartType.WaterFall;
//Set data range
chart.DataRange = chart.ChartData[1, 1, 8, 2];
chart.IsSeriesInRows = false;
chart.ChartData.SetValue(2, 1, "Start");
chart.ChartData.SetValue(2, 2, 120000);
chart.ChartData.SetValue(3, 1, "Product Revenue");
chart.ChartData.SetValue(3, 2, 570000);
chart.ChartData.SetValue(4, 1, "Service Revenue");
chart.ChartData.SetValue(4, 2, 230000);
chart.ChartData.SetValue(5, 1, "Positive Balance");
chart.ChartData.SetValue(5, 2, 920000);
chart.ChartData.SetValue(6, 1, "Fixed Costs");
chart.ChartData.SetValue(6, 2, -345000);
chart.ChartData.SetValue(7, 1, "Variable Costs");
chart.ChartData.SetValue(7, 2, -230000);
chart.ChartData.SetValue(8, 1, "Total");
chart.ChartData.SetValue(8, 2, 345000);
//Data point settings as total in chart
IOfficeChartSerie series = chart.Series[0];
chart.Series[0].DataPoints[2].SetAsTotal = true;
chart.Series[0].DataPoints[5].SetAsTotal = true;
//Showing the connector lines between data points
chart.Series[0].SerieFormat.ShowConnectorLines = true;
//Set the chart title
chart.ChartTitle = "Company Profit (in USD)";
//Formatting data label and legend option
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
chart.Legend.Position = OfficeLegendPosition.Right;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("WaterFall.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
```

XAMARIN

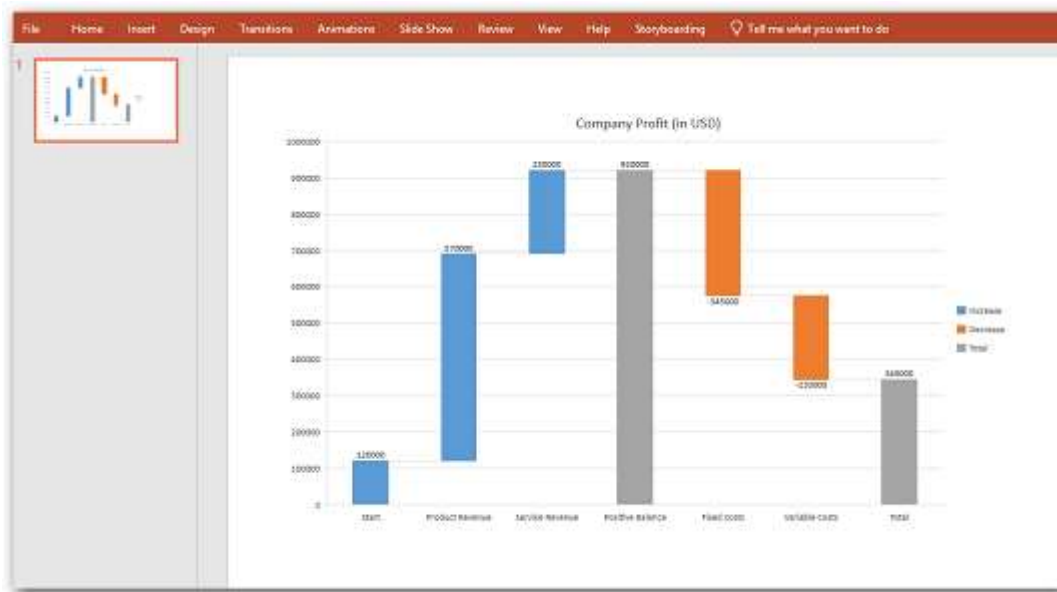
```
//Creates a new instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add slide to the presentation
ISlide slidel = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Create a chart
IPresentationChart chart = slidel.Charts.AddChart(50, 50, 700, 400);
//Set chart type as Waterfall
chart.ChartType = OfficeChartType.WaterFall;
```

```

//Set data range
chart.DataRange = chart.ChartData[1, 1, 8, 2];
chart.IsSeriesInRows = false;
chart.ChartData.SetValue(2, 1, "Start");
chart.ChartData.SetValue(2, 2, 120000);
chart.ChartData.SetValue(3, 1, "Product Revenue");
chart.ChartData.SetValue(3, 2, 570000);
chart.ChartData.SetValue(4, 1, "Service Revenue");
chart.ChartData.SetValue(4, 2, 230000);
chart.ChartData.SetValue(5, 1, "Positive Balance");
chart.ChartData.SetValue(5, 2, 920000);
chart.ChartData.SetValue(6, 1, "Fixed Costs");
chart.ChartData.SetValue(6, 2, -345000);
chart.ChartData.SetValue(7, 1, "Variable Costs");
chart.ChartData.SetValue(7, 2, -230000);
chart.ChartData.SetValue(8, 1, "Total");
chart.ChartData.SetValue(8, 2, 345000);
//Data point settings as total in chart
IOfficeChartSerie series = chart.Series[0];
chart.Series[0].DataPoints[2].SetAsTotal = true;
chart.Series[0].DataPoints[5].SetAsTotal = true;
//Showing the connector lines between data points
chart.Series[0].SerieFormat.ShowConnectorLines = true;
//Set the chart title
chart.ChartTitle = "Company Profit (in USD)";
//Formatting data label and legend option
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
chart.Legend.Position = OfficeLegendPosition.Right;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("WaterFall.ppt
x", "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("WaterFall.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

The following screenshot shows the output of previous code.



Histogram

[Histogram](#) shows the frequencies within a distribution. Each column of the chart is called a bin, which can be changed further to analyze the data. Refer to the following code example to create a Histogram.

C#

```
using (IPresentation pptxDoc = Presentation.Create())
{
    ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
    IPresentationChart chart = slide1.Charts.AddChart(50, 50, 500, 400);
    chart.ChartType = OfficeChartType.Histogram;
    chart.DataRange = chart.ChartData[2, 1, 15, 1];
    chart.ChartData.SetValue(1, 1, "Student Heights");
    chart.ChartData.SetValue(2, 1, 130);
    chart.ChartData.SetValue(3, 1, 132);
    chart.ChartData.SetValue(4, 1, 159);
    chart.ChartData.SetValue(5, 1, 163);
    chart.ChartData.SetValue(6, 1, 140);
    chart.ChartData.SetValue(7, 1, 155);
    chart.ChartData.SetValue(8, 1, 139);
    chart.ChartData.SetValue(9, 1, 143);
    chart.ChartData.SetValue(10, 1, 153);
    chart.ChartData.SetValue(11, 1, 165);
    chart.ChartData.SetValue(12, 1, 153);
    chart.ChartData.SetValue(13, 1, 149);
    chart.ChartData.SetValue(14, 1, 154);
    chart.ChartData.SetValue(15, 1, 162);
    //Category axis bin settings
    chart.PrimaryCategoryAxis.BinWidth = 8;
    //Gap width settings
    chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
    //Set the chart title and axis title
    chart.ChartTitle = "Height Data";
    chart.PrimaryValueAxis.Title = "Number of students";
    chart.PrimaryCategoryAxis.Title = "Height";
    //Hiding the legend
}
```

```

chart.HasLegend = false;
pptxDoc.Save("Histogram.pptx");
pptxDoc.Close();
}

```

VB.NET

```

'Creates a PowerPoint instance
Dim pptxDoc As IPresentation = Presentation.Create()
Dim slide1 As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
Dim chart As IPresentationChart = slide1.Charts.AddChart(50, 50, 500, 400)
chart.ChartType = OfficeChartType.Histogram
chart.DataRange = chart.ChartData(2, 1, 15, 1)
chart.ChartData.SetValue(1, 1, "Student Heights")
chart.ChartData.SetValue(2, 1, 130)
chart.ChartData.SetValue(3, 1, 132)
chart.ChartData.SetValue(4, 1, 159)
chart.ChartData.SetValue(5, 1, 163)
chart.ChartData.SetValue(6, 1, 140)
chart.ChartData.SetValue(7, 1, 155)
chart.ChartData.SetValue(8, 1, 139)
chart.ChartData.SetValue(9, 1, 143)
chart.ChartData.SetValue(10, 1, 153)
chart.ChartData.SetValue(11, 1, 165)
chart.ChartData.SetValue(12, 1, 153)
chart.ChartData.SetValue(13, 1, 149)
chart.ChartData.SetValue(14, 1, 154)
chart.ChartData.SetValue(15, 1, 162)
'Category axis bin settings
chart.PrimaryCategoryAxis.BinWidth = 8
'Gap width settings
chart.Series(0).SerieFormat.CommonSerieOptions.GapWidth = 6
'Set the chart title and axis title
chart.ChartTitle = "Height Data"
chart.PrimaryValueAxis.Title = "Number of students"
chart.PrimaryCategoryAxis.Title = "Height"
'Hiding the legend
chart.HasLegend = False
pptxDoc.Save("Histogram.pptx")
pptxDoc.Close()

```

UWP

```

//Creates a new instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
IPresentationChart chart = slide1.Charts.AddChart(50, 50, 500, 400);
chart.ChartType = OfficeChartType.Histogram;
chart.DataRange = chart.ChartData[2, 1, 15, 1];
chart.ChartData.SetValue(1, 1, "Student Heights");
chart.ChartData.SetValue(2, 1, 130);
chart.ChartData.SetValue(3, 1, 132);
chart.ChartData.SetValue(4, 1, 159);
chart.ChartData.SetValue(5, 1, 163);
chart.ChartData.SetValue(6, 1, 140);
chart.ChartData.SetValue(7, 1, 155);

```



```

chart.ChartData.SetValue(8, 1, 139);
chart.ChartData.SetValue(9, 1, 143);
chart.ChartData.SetValue(10, 1, 153);
chart.ChartData.SetValue(11, 1, 165);
chart.ChartData.SetValue(12, 1, 153);
chart.ChartData.SetValue(13, 1, 149);
chart.ChartData.SetValue(14, 1, 154);
chart.ChartData.SetValue(15, 1, 162);
//Category axis bin settings
chart.PrimaryCategoryAxis.BinWidth = 8;
//Gap width settings
chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
//Set the chart title and axis title
chart.ChartTitle = "Height Data";
chart.PrimaryValueAxis.Title = "Number of students";
chart.PrimaryCategoryAxis.Title = "Height";
//Hiding the legend
chart.HasLegend = false;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Histogram";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates a new instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
ISlide slidel = pptxDoc.Slides.Add(SlideLayoutType.Blank);
IPresentationChart chart = slidel.Charts.AddChart(50, 50, 500, 400);
chart.ChartType = OfficeChartType.Histogram;
chart.DataRange = chart.ChartData[2, 1, 15, 1];
chart.ChartData.SetValue(1, 1, "Student Heights");
chart.ChartData.SetValue(2, 1, 130);
chart.ChartData.SetValue(3, 1, 132);
chart.ChartData.SetValue(4, 1, 159);
chart.ChartData.SetValue(5, 1, 163);
chart.ChartData.SetValue(6, 1, 140);
chart.ChartData.SetValue(7, 1, 155);
chart.ChartData.SetValue(8, 1, 139);
chart.ChartData.SetValue(9, 1, 143);
chart.ChartData.SetValue(10, 1, 153);
chart.ChartData.SetValue(11, 1, 165);
chart.ChartData.SetValue(12, 1, 153);
chart.ChartData.SetValue(13, 1, 149);
chart.ChartData.SetValue(14, 1, 154);
chart.ChartData.SetValue(15, 1, 162);
//Category axis bin settings
chart.PrimaryCategoryAxis.BinWidth = 8;
//Gap width settings
chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;

```

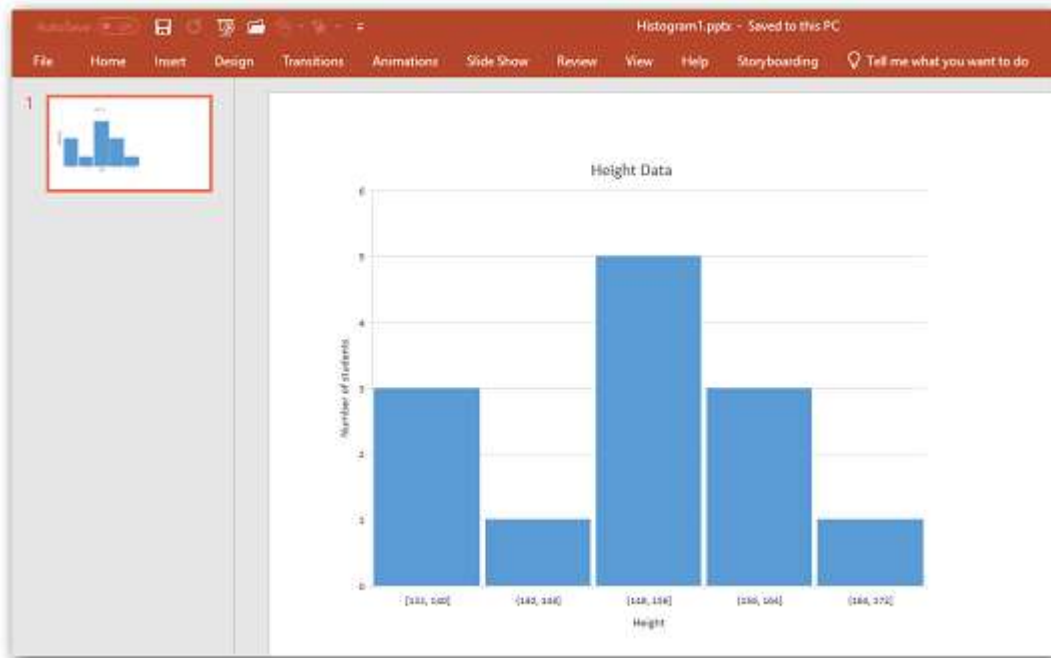
```
//Set the chart title and axis title
chart.ChartTitle = "Height Data";
chart.PrimaryValueAxis.Title = "Number of students";
chart.PrimaryCategoryAxis.Title = "Height";
//Hiding the legend
chart.HasLegend = false;
pptxDoc.Save("Histogram.pptx");
pptxDoc.Close();
```

XAMARIN

```
//Creates a new instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
ISlide slidel = pptxDoc.Slides.Add(SlideLayoutType.Blank);
IPresentationChart chart = slidel.Charts.AddChart(50, 50, 500, 400);
chart.ChartType = OfficeChartType.Histogram;
chart.DataRange = chart.ChartData[2, 1, 15, 1];
chart.ChartData.SetValue(1, 1, "Student Heights");
chart.ChartData.SetValue(2, 1, 130);
chart.ChartData.SetValue(3, 1, 132);
chart.ChartData.SetValue(4, 1, 159);
chart.ChartData.SetValue(5, 1, 163);
chart.ChartData.SetValue(6, 1, 140);
chart.ChartData.SetValue(7, 1, 155);
chart.ChartData.SetValue(8, 1, 139);
chart.ChartData.SetValue(9, 1, 143);
chart.ChartData.SetValue(10, 1, 153);
chart.ChartData.SetValue(11, 1, 165);
chart.ChartData.SetValue(12, 1, 153);
chart.ChartData.SetValue(13, 1, 149);
chart.ChartData.SetValue(14, 1, 154);
chart.ChartData.SetValue(15, 1, 162);
//Category axis bin settings
chart.PrimaryCategoryAxis.BinWidth = 8;
//Gap width settings
chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
//Set the chart title and axis title
chart.ChartTitle = "Height Data";
chart.PrimaryValueAxis.Title = "Number of students";
chart.PrimaryCategoryAxis.Title = "Height";
//Hiding the legend
chart.HasLegend = false;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Histogram.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Histogram.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

The following screenshot shows the output of previous code.



Pareto

[Pareto](#) is a sorted histogram in which the columns sorted in descending order and a line representing the cumulative total percentage. . Refer to the following code example to create a Pareto chart.

C#

```
using(IPresentation pptxDoc = Presentation.Create())
{
    ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
    IPresentationChart chart = slide1.Charts.AddChart(50, 50, 500, 400);
    //Set chart type as Pareto
    chart.ChartType = OfficeChartType.Pareto;
    //Set data range
    chart.DataRange = chart.ChartData[2, 1, 8, 2];
    chart.ChartData.SetValue(2, 1, "Rent");
    chart.ChartData.SetValue(2, 2, 2300);
    chart.ChartData.SetValue(3, 1, "Car payment");
    chart.ChartData.SetValue(3, 2, 1200);
    chart.ChartData.SetValue(4, 1, "Groceries");
    chart.ChartData.SetValue(4, 2, 900);
    chart.ChartData.SetValue(5, 1, "Electricity");
    chart.ChartData.SetValue(5, 2, 600);
    chart.ChartData.SetValue(6, 1, "Gas");
```

```

chart.ChartData.SetValue(6, 2, 500);
chart.ChartData.SetValue(7, 1, "Cable");
chart.ChartData.SetValue(7, 2, 300);
chart.ChartData.SetValue(8, 1, "Mobile");
chart.ChartData.SetValue(8, 2, 200);
//Set category values as bin values
chart.PrimaryCategoryAxis.IsBinningByCategory = true;
//Formatting Pareto line
chart.Series[0].ParetoLineFormat.LineProperties.ColorIndex =
OfficeKnownColors.Bright_green;
//Gap width settings
chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
//Set the chart title
chart.ChartTitle = "Expenses";
//Hiding the legend
chart.HasLegend = false;
//Save and close the presentation
pptxDoc.Save("ParetoChart.pptx");
pptxDoc.Close();
}

```

VB.NET

```

'Creates a PowerPoint instance
Dim pptxDoc As IPresentation = Presentation.Create()
Dim slide1 As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
Dim chart As IPresentationChart = slide1.Charts.AddChart(50, 50, 500, 400)
'Set chart type as Pareto
chart.ChartType = OfficeChartType.Pareto
'Set data range
chart.DataRange = chart.ChartData(2, 1, 8, 2)
chart.ChartData.SetValue(2, 1, "Rent")
chart.ChartData.SetValue(2, 2, 2300)
chart.ChartData.SetValue(3, 1, "Car payment")
chart.ChartData.SetValue(3, 2, 1200)
chart.ChartData.SetValue(4, 1, "Groceries")
chart.ChartData.SetValue(4, 2, 900)
chart.ChartData.SetValue(5, 1, "Electricity")
chart.ChartData.SetValue(5, 2, 600)
chart.ChartData.SetValue(6, 1, "Gas")
chart.ChartData.SetValue(6, 2, 500)
chart.ChartData.SetValue(7, 1, "Cable")
chart.ChartData.SetValue(7, 2, 300)
chart.ChartData.SetValue(8, 1, "Mobile")
chart.ChartData.SetValue(8, 2, 200)
'Set category values as bin values
chart.PrimaryCategoryAxis.IsBinningByCategory = True
'Formatting Pareto line
chart.Series(0).ParetoLineFormat.LineProperties.ColorIndex =
OfficeKnownColors.Bright_green
'Gap width settings
chart.Series(0).SerieFormat.CommonSerieOptions.GapWidth = 6
'Set the chart title
chart.ChartTitle = "Expenses"
'Hiding the legend
chart.HasLegend = False

```

```
'Save and close the presentation
pptxDoc.Save("ParetoChart.pptx")
pptxDoc.Close()
```

UWP

```
//Creates a new instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a slide to Presentation
ISlide slidel = pptxDoc.Slides.Add(SlideLayoutType.Blank);
IPresentationChart chart = slidel.Charts.AddChart(50, 50, 500, 400);
//Set chart type as Pareto
chart.ChartType = OfficeChartType.Pareto;
//Set data range
chart.DataRange = chart.ChartData[2, 1, 8, 2];
chart.ChartData.SetValue(2, 1, "Rent");
chart.ChartData.SetValue(2, 2, 2300);
chart.ChartData.SetValue(3, 1, "Car payment");
chart.ChartData.SetValue(3, 2, 1200);
chart.ChartData.SetValue(4, 1, "Groceries");
chart.ChartData.SetValue(4, 2, 900);
chart.ChartData.SetValue(5, 1, "Electricity");
chart.ChartData.SetValue(5, 2, 600);
chart.ChartData.SetValue(6, 1, "Gas");
chart.ChartData.SetValue(6, 2, 500);
chart.ChartData.SetValue(7, 1, "Cable");
chart.ChartData.SetValue(7, 2, 300);
chart.ChartData.SetValue(8, 1, "Mobile");
chart.ChartData.SetValue(8, 2, 200);
//Set category values as bin values
chart.PrimaryCategoryAxis.IsBinningByCategory = true;
//Formatting Pareto line
chart.Series[0].ParetoLineFormat.LineProperties.ColorIndex =
OfficeKnownColors.Bright_green;
//Gap width settings
chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
//Set the chart title
chart.ChartTitle = "Expenses";
//Hiding the legend
chart.HasLegend = false;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ParetoChart";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Creates a new instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a slide to Presentation
```

```

ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
IPresentationChart chart = slide1.Charts.AddChart(50, 50, 500, 400);
//Set chart type as Pareto
chart.ChartType = OfficeChartType.Pareto;
//Set data range
chart.DataRange = chart.ChartData[2, 1, 8, 2];
chart.ChartData.SetValue(2, 1, "Rent");
chart.ChartData.SetValue(2, 2, 2300);
chart.ChartData.SetValue(3, 1, "Car payment");
chart.ChartData.SetValue(3, 2, 1200);
chart.ChartData.SetValue(4, 1, "Groceries");
chart.ChartData.SetValue(4, 2, 900);
chart.ChartData.SetValue(5, 1, "Electricity");
chart.ChartData.SetValue(5, 2, 600);
chart.ChartData.SetValue(6, 1, "Gas");
chart.ChartData.SetValue(6, 2, 500);
chart.ChartData.SetValue(7, 1, "Cable");
chart.ChartData.SetValue(7, 2, 300);
chart.ChartData.SetValue(8, 1, "Mobile");
chart.ChartData.SetValue(8, 2, 200);
//Set category values as bin values
chart.PrimaryCategoryAxis.IsBinningByCategory = true;
//Formatting Pareto line
chart.Series[0].ParetoLineFormat.LineProperties.ColorIndex =
OfficeKnownColors.Bright_green;
//Gap width settings
chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
//Set the chart title
chart.ChartTitle = "Expenses";
//Hiding the legend
chart.HasLegend = false;
//Save and close the presentation
pptxDoc.Save("ParetoChart.pptx");
pptxDoc.Close();

```

XAMARIN

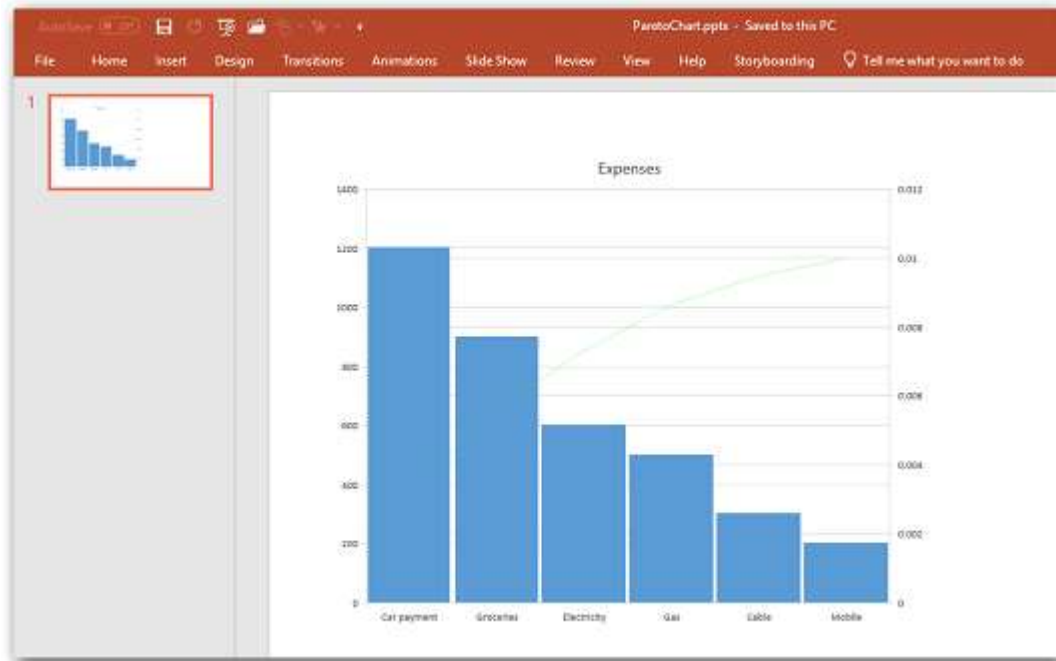
```

//Creates a new instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a slide to Presentation
ISlide slide1 = pptxDoc.Slides.Add(SlideLayoutType.Blank);
IPresentationChart chart = slide1.Charts.AddChart(50, 50, 500, 400);
//Set chart type as Pareto
chart.ChartType = OfficeChartType.Pareto;
//Set data range
chart.DataRange = chart.ChartData[2, 1, 8, 2];
chart.ChartData.SetValue(2, 1, "Rent");
chart.ChartData.SetValue(2, 2, 2300);
chart.ChartData.SetValue(3, 1, "Car payment");
chart.ChartData.SetValue(3, 2, 1200);
chart.ChartData.SetValue(4, 1, "Groceries");
chart.ChartData.SetValue(4, 2, 900);
chart.ChartData.SetValue(5, 1, "Electricity");
chart.ChartData.SetValue(5, 2, 600);
chart.ChartData.SetValue(6, 1, "Gas");
chart.ChartData.SetValue(6, 2, 500);

```

```
chart.ChartData.SetValue(7, 1, "Cable");
chart.ChartData.SetValue(7, 2, 300);
chart.ChartData.SetValue(8, 1, "Mobile");
chart.ChartData.SetValue(8, 2, 200);
//Set category values as bin values
chart.PrimaryCategoryAxis.IsBinningByCategory = true;
//Formatting Pareto line
chart.Series[0].ParetoLineFormat.LineProperties.ColorIndex =
OfficeKnownColors.Bright_green;
//Gap width settings
chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
//Set the chart title
chart.ChartTitle = "Expenses";
//Hiding the legend
chart.HasLegend = false;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("ParetoChart.p
ptx", "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("ParetoChart.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

The following screenshot shows the output of previous code.



Note: These charts are supported only in PowerPoint 2016 and are not supported in the previous versions.

Supported Chart Types

The following Chart types are supported in Presentation.

- Area
- Area_3D
- Area_Stacked
- AreaStacked100
- AreaStacked100_3D
- AreaStacked3D
- Bar_Clustered
- BarClustered3D
- Bar_Stacked
- BarStacked100
- BarStacked100_3D
- BarStacked3D
- Bubble
- Bubble_3D
- Column_3D
- Column_Clustered
- ColumnClustered3D
- Column_Stacked
- ColumnStacked100
- ColumnStacked100_3D
- ColumnStacked3D
- Combination_Chart
- ConeBarClustered

- ConeBarStacked
- ConeBarStacked_100
- Cone_Clustered
- ConeClustered3D
- Cone_Stacked
- ConeStacked100
- CylinderBarClustered
- CylinderBarStacked
- CylinderBarStacked_100
- Cylinder_Clustered
- CylinderClustered3D
- Cylinder_Stacked
- CylinderStacked100
- Doughnut
- Doughnut_Exploded
- Line
- Line_3D
- Line_Markers
- LineMarkersStacked
- LineMarkersStacked_100
- Line_Stacked
- LineStacked100
- Pie
- Pie_3D
- Pie_Bar
- Pie_Exploded
- PieExploded3D
- PieOfPie
- PyramidBarClustered
- PyramidBarStacked
- PyramidBarStacked_100
- Pyramid_Clustered
- PyramidClustered3D
- Pyramid_Stacked
- PyramidStacked100
- Radar
- Radar_Filled
- Radar_Markers
- Scatter_Line
- ScatterLineMarkers
- Scatter_Markers
- Scatter_SmoothedLine
- ScatterSmoothedLineMarkers
- Stock_HighLowClose
- Stock_OpenHighLowClose
- Stock_VolumeHighLowClose
- Stock_VolumeOpenHighLowClose
- Surface_3D

- Surface_Contour
- SurfaceNoColor3D
- SurfaceNoColorContour

Working with Animations

Animations are visual effects for the objects in PowerPoint presentation and animation helps to make a PowerPoint presentation more dynamic. Animation effects can be grouped into four categories.,

1. Entrance
2. Emphasis
3. Exit
4. Motion paths

Entrance effects can be set to enter the objects with animations during slide show. Emphasis effects animate the objects on the spot. Exit effects allow objects to leave the slide show with animations. Motion Paths allow objects to move around the slide show. Each effect contains variables such as start (On click, with previous and after previous), delay, speed, repeat, and trigger. This makes animations more flexible and interactive.

Syncfusion Presentation library allows you to animate the text, pictures, shapes, tables, SmartArt graphics, and charts in PowerPoint presentation.

Adding animation effect to shapes

Animation effects can be added to shapes, images, tables, charts and SmartArt diagrams. The following code example demonstrates how to add an animation effect to a shape.

C#

```
//Create an instance for PowerPoint
using (IPresentation pptxDoc = Presentation.Create())
{
    //Add a blank slide to Presentation
    ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
    //Add normal shape to slide
    IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300);
    //Access the animation sequence to create effects
    ISequence sequence = slide.Timeline.MainSequence;
    //Add bounce effect to the shape
    IEffect bounceEffect = sequence.AddEffect(cubeShape, EffectType.Bounce, EffectSubtype.None, EffectTriggerType.OnClick);
    //Save the Presentation
    pptxDoc.Save("Sample.pptx");
}
```

VB.NET

```
'Create an instance for PowerPoint
Using pptxDoc As IPresentation = Presentation.Create()
'Add a blank slide to Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add normal shape to slide
```

```

Dim cubeShape As IShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200,
300, 300)
'Access the animation sequence to create effects
Dim sequence As ISequence = slide.Timeline.MainSequence
'Add bounce effect to the shape
Dim bounceEffect As IEffect = sequence.AddEffect(cubeShape,
EffectType.Bounce, EffectSubtype.None, EffectTriggerType.OnClick)
'Save the Presentation
pptxDoc.Save("Sample.pptx")
End Using

```

UWP

```

//Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add normal shape to slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add bounce effect to the shape
IEffect bounceEffect = sequence.AddEffect(cubeShape, EffectType.Bounce,
EffectSubtype.None, EffectTriggerType.OnClick);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Create an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add normal shape to slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add bounce effect to the shape
IEffect bounceEffect = sequence.AddEffect(cubeShape, EffectType.Bounce,
EffectSubtype.None, EffectTriggerType.OnClick);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);

```

XAMARIN

```

//Create an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add normal shape to slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add bounce effect to the shape
IEffect bounceEffect = sequence.AddEffect(cubeShape, EffectType.Bounce,
EffectSubtype.None, EffectTriggerType.OnClick);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Adding interactive animation

Animations can be interactive when it depends on another slide element., for example, an animation associated with a rectangle is triggered when user clicks an oval shape in the slide. The following code example demonstrates how to set an interactive animation.

C#

```

//Create an instance for PowerPoint
using (IPresentation pptxDoc= Presentation.Create())
{
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add normal shape to slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Add a shape to act as button
IShape buttonShape = slide.Shapes.AddShape(AutoShapeType.Oval,
100,100,50,50);
//Create the interactive sequence to make the animation effects interactive
by triggering with button click
ISequence interactiveSequence =
slide.Timeline.InteractiveSequences.Add(buttonShape);
//Add Fly effect with top subtype to animate the shape as fly from top

```

```
IEffect bounceEffect = interactiveSequence.AddEffect(cubeShape,
EffectType.Fly, EffectSubtype.Top, EffectTriggerType.OnClick);
//Save the Presentation
pptxDoc.Save("Sample.pptx");
}
```

VB.NET

```
'Create an instance for PowerPoint
Using pptxDoc As IPresentation = Presentation.Create()
'Add a blank slide to Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add normal shape to slide
Dim cubeShape As IShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200,
300, 300)
'Add a shape to act as button
Dim buttonShape As IShape = slide.Shapes.AddShape(AutoShapeType.Oval, 100,
100, 50, 50)
'Create the interactive sequence to make the animation effects interactive
by triggering with button click
Dim interactiveSequence As ISequence =
slide.Timeline.InteractiveSequences.Add(buttonShape)
'Add Fly effect with top subtype to animate the shape as fly from top
Dim bounceEffect As IEffect = interactiveSequence.AddEffect(cubeShape,
EffectType.Fly, EffectSubtype.Top, EffectTriggerType.OnClick)
'Save the Presentation
pptxDoc.Save("Sample.pptx")
End Using
```

UWP

```
//Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add normal shape to slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Add a shape to act as button
IShape buttonShape = slide.Shapes.AddShape(AutoShapeType.Oval,
100,100,50,50);
//Create the interactive sequence to make the animation effects interactive
by triggering with button click
ISequence interactiveSequence =
slide.Timeline.InteractiveSequences.Add(buttonShape);
//Add Fly effect with top subtype to animate the shape as fly from top
IEffect bounceEffect = interactiveSequence.AddEffect(cubeShape,
EffectType.Fly, EffectSubtype.Top, EffectTriggerType.OnClick);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
```

```
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Create an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add normal shape to slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Add a shape to act as button
IShape buttonShape = slide.Shapes.AddShape(AutoShapeType.Oval,
100,100,50,50);
//Create the interactive sequence to make the animation effects interactive
by triggering with button click
ISequence interactiveSequence =
slide.Timeline.InteractiveSequences.Add(buttonShape);
//Add Fly effect with top subtype to animate the shape as fly from top
IEffect bounceEffect = interactiveSequence.AddEffect(cubeShape,
EffectType.Fly, EffectSubtype.Top, EffectTriggerType.OnClick);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
```

XAMARIN

```
//Create an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add normal shape to slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Add a shape to act as button
IShape buttonShape = slide.Shapes.AddShape(AutoShapeType.Oval,
100,100,50,50);
//Create the interactive sequence to make the animation effects interactive
by triggering with button click
ISequence interactiveSequence =
slide.Timeline.InteractiveSequences.Add(buttonShape);
//Add Fly effect with top subtype to animate the shape as fly from top
IEffect bounceEffect = interactiveSequence.AddEffect(cubeShape,
EffectType.Fly, EffectSubtype.Top, EffectTriggerType.OnClick);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Adding animation to text

Animation effects can be applied to text. The following code example demonstrated how to set an animation effect to a text.

C#

```

//Open an existing Presentation from file system
using (IPresentation pptxDoc = Presentation.Open("Sample.pptx"))
{
//Retrieve the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieve the first shape
IShape shape = slide.Shapes[0] as IShape;
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add swivel effect with vertical subtype to the shape, build type is used
to represent the animate level of the paragraph
IEffect bounceEffect = sequence.AddEffect(shape, EffectType.Swivel,
EffectSubtype.Vertical, EffectTriggerType.OnClick,
BuildType.ByLevelParagraphs1);
//Save the Presentation to the file system
pptxDoc.Save("Result.pptx");
}

```

VB.NET

```

'Opens an existing Presentation from file system
Using pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Retrieve the first slide from Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Retrieve the first shape
Dim shape As IShape = TryCast(slide.Shapes(0), IShape)
'Access the animation sequence to create effects
Dim sequence As ISequence = slide.Timeline.MainSequence
'Add swivel effect with vertical subtype to the shape, build type is used to
represent the animate level of the paragraph
Dim bounceEffect As IEffect = sequence.AddEffect(shape, EffectType.Swivel,
EffectSubtype.Vertical, EffectTriggerType.OnClick,
BuildType.ByLevelParagraphs1)
'Save the Presentation to the file system
pptxDoc.Save("Result.pptx")
End Using

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieve the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieve the first shape
IShape shape = slide.Shapes[0] as IShape;
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add swivel effect with vertical subtype to the shape, build type is used to represent the animate level of the paragraph
IEffect bounceEffect = sequence.AddEffect(shape, EffectType.Swivel, EffectSubtype.Vertical, EffectTriggerType.OnClick, BuildType.ByLevelParagraphs1);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() { ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieve the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieve the first shape
IShape shape = slide.Shapes[0] as IShape;
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add swivel effect with vertical subtype to the shape, build type is used to represent the animate level of the paragraph
IEffect bounceEffect = sequence.AddEffect(shape, EffectType.Swivel, EffectSubtype.Vertical, EffectTriggerType.OnClick, BuildType.ByLevelParagraphs1);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Result.pptx", FileMode.Create);
pptxDoc.Save(outputStream);

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;

```



```

Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieve the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieve the first shape
IShape shape = slide.Shapes[0] as IShape;
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add swivel effect with vertical subtype to the shape, build type is used
to represent the animate level of the paragraph
IEffect bounceEffect = sequence.AddEffect(shape, EffectType.Swivel,
EffectSubtype.Vertical, EffectTriggerType.OnClick,
BuildType.ByLevelParagraphs1);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Result.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Result.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Adding exit animation effect

When you add common animation effects for both entrance and exit types, animation is applied with entrance effect by default. The following code example demonstrates how to set exist type animation for a shape.

C#

```

//Create an instance for PowerPoint
using (IPresentation pptxDoc = Presentation.Create())
{
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add normal shape to slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add random bars effect to the shape
IEffect effect = sequence.AddEffect(cubeShape, EffectType.RandomBars,
EffectSubtype.None, EffectTriggerType.OnClick);

```

```
//Change the preset class type of the effect from default entrance to exit
effect.PresetClassType = EffectPresetClassType.Exit;
//Save the Presentation
pptxDoc.Save("Sample.pptx");
}
```

VB.NET

```
'Create an instance for PowerPoint
Using pptxDoc As IPresentation = Presentation.Create()
'Add a blank slide to Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add normal shape to slide
Dim cubeShape As IShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200,
300, 300)
'Access the animation sequence to create effects
Dim sequence As ISequence = slide.Timeline.MainSequence
'Add random bars effect to the shape
Dim effect As IEfffect = sequence.AddEffect(cubeShape, EffectType.RandomBars,
EffectSubtype.None, EffectTriggerType.OnClick)
'Change the preset class type of the effect from default entrance to exit
effect.PresetClassType = EffectPresetClassType.[Exit]
'Save the Presentation
pptxDoc.Save("Sample.pptx")
End Using
```

UWP

```
//Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add normal shape to slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add random bars effect to the shape
IEfffect effect = sequence.AddEffect(cubeShape, EffectType.RandomBars,
EffectSubtype.None, EffectTriggerType.OnClick);
//Change the preset class type of the effect from default entrance to exit
effect.PresetClassType = EffectPresetClassType.Exit;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Create an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add normal shape to slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300);
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add random bars effect to the shape
IEffect effect = sequence.AddEffect(cubeShape, EffectType.RandomBars, EffectSubtype.None, EffectTriggerType.OnClick);
//Change the preset class type of the effect from default entrance to exit
effect.PresetClassType = EffectPresetClassType.Exit;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
```

XAMARIN

```
//Create an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add normal shape to slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300);
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add random bars effect to the shape
IEffect effect = sequence.AddEffect(cubeShape, EffectType.RandomBars, EffectSubtype.None, EffectTriggerType.OnClick);
//Change the preset class type of the effect from default entrance to exit
effect.PresetClassType = EffectPresetClassType.Exit;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
```

Edit existing animation effect

The Presentation library allows you to edit the animations in existing presentations. The following example demonstrates how to modify an existing animation applied to a shape.

C#

```
//Open an existing Presentation from file system
using (IPresentation pptxDoc = Presentation.Open("Sample.pptx"))
{
    //Retrieve the first slide from Presentation
    ISlide slide = pptxDoc.Slides[0];
    //Retrieve the first shape
    IShape shape = slide.Shapes[0] as IShape;
    //Access the animation main sequence to modify the effects
    ISequence sequence = slide.Timeline.MainSequence;
    //Get the animation effects of the particular shape
    IEffect[] animationEffects = sequence.GetEffectsByShape(shape);
    //Iterate the animation effect to make the change
    IEffect animationEffect = animationEffects[0];
    //Change the animation effect type from swivel to GrowAndTurn
    animationEffect.Type = EffectType.GrowAndTurn;
    //Save the Presentation to the file system
    pptxDoc.Save("Animation.pptx");
}
```

VB.NET

```
'Open an existing Presentation from file system
Using pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Retrieve the first slide from Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Retrieve the first shape.
Dim shape As IShape = TryCast(slide.Shapes(0), IShape)
'Access the animation main sequence to modify the effects
Dim sequence As ISequence = slide.Timeline.MainSequence
'Get the animation effects of the particular shape
Dim animationEffects As IEffect() = sequence.GetEffectsByShape(shape)
'Iterate the animation effect to make the change
Dim animationEffect As IEffect = animationEffects(0)
'Change the animation effect type from swivel to GrowAndTurn
animationEffect.Type = EffectType.GrowAndTurn
'Save the Presentation to the file system
pptxDoc.Save("Animation.pptx")
End Using
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieve the first slide from Presentation
```

```

ISlide slide = pptxDoc.Slides[0];
//Retrieve the first shape
IShape shape = slide.Shapes[0] as IShape;
//Access the animation main sequence to modify the effects
ISequence sequence = slide.Timeline.MainSequence;
//Get the animation effects of the particular shape
IEffect[] animationEffects = sequence.GetEffectsByShape(shape);
//Iterate the animation effect to make the change
IEffect animationEffect = animationEffects[0];
//Change the animation effect type from swivel to GrowAndTurn
animationEffect.Type = EffectType.GrowAndTurn;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieve the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieve the first shape
IShape shape = slide.Shapes[0] as IShape;
//Access the animation main sequence to modify the effects
ISequence sequence = slide.Timeline.MainSequence;
//Get the animation effects of the particular shape
IEffect[] animationEffects = sequence.GetEffectsByShape(shape);
//Iterate the animation effect to make the change
IEffect animationEffect = animationEffects[0];
//Change the animation effect type from swivel to GrowAndTurn
animationEffect.Type = EffectType.GrowAndTurn;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Animation.pptx", FileMode.Create);
pptxDoc.Save(outputStream);

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieve the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieve the first shape
IShape shape = slide.Shapes[0] as IShape;

```

```

//Access the animation main sequence to modify the effects
ISequence sequence = slide.Timeline.MainSequence;
//Get the animation effects of the particular shape
IEffect[] animationEffects = sequence.GetEffectsByShape(shape);
//Iterate the animation effect to make the change
IEffect animationEffect = animationEffects[0];
//Change the animation effect type from swivel to GrowAndTurn
animationEffect.Type = EffectType.GrowAndTurn;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Animation.ppt
x", "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Animation.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Modifying animation effect sub type

Presentation library allows you to edit the sub type of animations effects in existing presentations. The following example demonstrates how to modify a sub type applied to the existing animation.

C#

```

//Opens an existing Presentation from file system
using (IPresentation pptxDoc = Presentation.Open("Sample.pptx"))
{
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape
IShape shape = slide.Shapes[0] as IShape;
//Access the animation main sequence to modify the effects
ISequence sequence = slide.Timeline.MainSequence;
//Get the required animation effect from the slide
IEffect wheelEffect = sequence[0] as IEffect;
//Change the wheel animation effect sub type from 2 spoke to 4 spoke
wheelEffect.Subtype = EffectSubtype.Wheel4;
//Saves the Presentation to the file system
pptxDoc.Save("Result.pptx");
}

```

VB.NET

```

'Opens an existing Presentation from file system
Using pptxDoc As IPresentation = Presentation.Open("Sample.pptx")

```

```

'Retrieves the first slide from Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Retrieves the first shape
Dim shape As IShape = TryCast(slide.Shapes(0), IShape)
'Access the animation main sequence to modify the effects
Dim sequence As ISequence = slide.Timeline.MainSequence
'Get the required animation effect from the slide
Dim wheelEffect As IEffect = TryCast(sequence(0), IEffect)
'Change the wheel animation effect sub type from 2 spoke to 4 spoke
wheelEffect.Subtype = EffectSubtype.Wheel4
'Saves the Presentation to the file system
pptxDoc.Save("Result.pptx")
End Using

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape
IShape shape = slide.Shapes[0] as IShape;
//Access the animation main sequence to modify the effects
ISequence sequence = slide.Timeline.MainSequence;
//Get the required animation effect from the slide
IEffect wheelEffect = sequence[0] as IEffect;
//Change the wheel animation effect sub type from 2 spoke to 4 spoke
wheelEffect.Subtype = EffectSubtype.Wheel4;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Result";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape
IShape shape = slide.Shapes[0] as IShape;
//Access the animation main sequence to modify the effects
ISequence sequence = slide.Timeline.MainSequence;

```

```
//Get the required animation effect from the slide
IEffect wheelEffect = sequence[0] as IEffect;
//Change the wheel animation effect sub type from 2 spoke to 4 spoke
wheelEffect.Subtype = EffectSubtype.Wheel4;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Animation.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
```

XAMARIN

```
///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape
IShape shape = slide.Shapes[0] as IShape;
//Access the animation main sequence to modify the effects
ISequence sequence = slide.Timeline.MainSequence;
//Get the required animation effect from the slide
IEffect wheelEffect = sequence[0] as IEffect;
//Change the wheel animation effect sub type from 2 spoke to 4 spoke
wheelEffect.Subtype = EffectSubtype.Wheel4;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Animation.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Animation.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Modifying timing of animation effect

Presentation library allows you to edit the animation timing in the existing presentations. The following example demonstrates how to modify an existing animation timing applied to a shape.

C#

```
//Open an existing Presentation from file system
using (IPresentation pptxDoc = Presentation.Open("Sample.pptx"))
```



```

{
    //Retrieves the first slide from Presentation
    ISlide slide = pptxDoc.Slides[0];
    //Retrieves the first shape
    IShape shape = slide.Shapes[0] as IShape;
    //Access the animation main sequence to modify the effects
    ISequence sequence = slide.Timeline.MainSequence;
    //Get the required animation effect from the slide
    IEffect pathEffect = sequence[0] as IEffect;
    //Increase the duration of the animation effect
    pathEffect.Behaviors[0].Timing.Duration = 5;
    //Saves the Presentation to the file system
    pptxDoc.Save("Result.pptx");
}

```

VB.NET

```

'Open an existing Presentation from file system
Using pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Retrieves the first slide from Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Retrieves the first shape
Dim shape As IShape = TryCast(slide.Shapes(0), IShape)
'Access the animation main sequence to modify the effects
Dim sequence As ISequence = slide.Timeline.MainSequence
'Get the required animation effect from the slide
Dim pathEffect As IEffect = TryCast(sequence(0), IEffect)
'Increase the duration of the animation effect
pathEffect.Behaviors(0).Timing.Duration = 5
'Save the Presentation to the file system.
pptxDoc.Save("Result.pptx")
End Using

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = await Presentation.OpenAsync(inputStorageFile);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape
IShape shape = slide.Shapes[0] as IShape;
//Access the animation main sequence to modify the effects
ISequence sequence = slide.Timeline.MainSequence;
//Get the required animation effect from the slide
IEffect pathEffect = sequence[0] as IEffect;
//Increase the duration of the animation effect
pathEffect.Behaviors[0].Timing.Duration = 5;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;

```

```

savePicker.SuggestedFileName = "Result";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape
IShape shape = slide.Shapes[0] as IShape;
//Access the animation main sequence to modify the effects
ISequence sequence = slide.Timeline.MainSequence;
//Get the required animation effect from the slide
IEffect pathEffect = sequence[0] as IEffect;
//Increase the duration of the animation effect
pathEffect.Behaviors[0].Timing.Duration = 5;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Animation.pptx", FileMode.Create);
pptxDoc.Save(outputStream);

```

XAMARIN

```

/"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieves the first slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape
IShape shape = slide.Shapes[0] as IShape;
//Access the animation main sequence to modify the effects
ISequence sequence = slide.Timeline.MainSequence;
//Get the required animation effect from the slide
IEffect pathEffect = sequence[0] as IEffect;
//Increase the duration of the animation effect
pathEffect.Behaviors[0].Timing.Duration = 5;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Animation.ppt
x", "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Animation.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Reordering the animation effects

Presentation library allows you to reorder the animation effects in existing presentations. The following example demonstrates how to modify an existing animation order applied to a shape.

C#

```

//Open the existing presentation
using (IPresentation pptxDoc = Presentation.Open("Sample.pptx"))
{
    //Iterate the slide
    ISlide slide = pptxDoc.Slides[0];
    //Iterate the shape
    IShape shape = slide.Shapes[0] as IShape;
    //Iterate the sequence
    ISequence sequence = slide.Timeline.MainSequence;
    //Get the animation effects of the shape
    IEffect[] shapeAnimationEffects = sequence.GetEffectsByShape(shape);
    //Get the second animation effect of the shape
    IEffect effect = shapeAnimationEffects[1];
    //Remove the animation effect from the sequence
    sequence.Remove(effect);
    //Insert the removed animation effect as first
    sequence.Insert(0, effect);
    //Save the created presentation
    pptxDoc.Save("Output.pptx");
}

```

VB.NET

```

'Open the existing presentation
Using pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
    'Iterate the slide
    Dim slide As ISlide = pptxDoc.Slides(0)
    'Iterate the shape
    Dim shape As IShape = TryCast(slide.Shapes(0), IShape)
    'Iterate the sequence
    Dim sequence As ISequence = slide.Timeline.MainSequence
    'Get the animation effects of the shape
    Dim shapeAnimationEffects As IEffect() = sequence.GetEffectsByShape(shape)
    'Get the second animation effect of the shape
    Dim effect As IEffect = shapeAnimationEffects(1)
    'Remove the animation effect from the sequence
    sequence.Remove(effect)
    'Insert the removed animation effect as first
    sequence.Insert(0, effect)

```

```
'Save the created presentation
pptxDoc.Save("Output.pptx")
End Using
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Iterate the slide
ISlide slide = pptxDoc.Slides[0];
//Iterate the shape
IShape shape = slide.Shapes[0] as IShape;
//Iterate the sequence
ISequence sequence = slide.Timeline.MainSequence;
//Get the animation effects of the shape
IEffect[] shapeAnimationEffects = sequence.GetEffectsByShape(shape);
//Get the second animation effect of the shape
IEffect effect = shapeAnimationEffects[1];
//Remove the animation effect from the sequence
sequence.Remove(effect);
//Insert the removed animation effect as first
sequence.Insert(0, effect);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Iterate the slide
ISlide slide = pptxDoc.Slides[0];
//Iterate the shape
IShape shape = slide.Shapes[0] as IShape;
//Iterate the sequence
ISequence sequence = slide.Timeline.MainSequence;
//Get the animation effects of the shape
IEffect[] shapeAnimationEffects = sequence.GetEffectsByShape(shape);
//Get the second animation effect of the shape
IEffect effect = shapeAnimationEffects[1];
//Remove the animation effect from the sequence
sequence.Remove(effect);
```

```
//Insert the removed animation effect as first
sequence.Insert(0, effect);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Animation.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Iterate the slide
ISlide slide = pptxDoc.Slides[0];
//Iterate the shape
IShape shape = slide.Shapes[0] as IShape;
//Iterate the sequence
ISequence sequence = slide.Timeline.MainSequence;
//Get the animation effects of the shape
IEffect[] shapeAnimationEffects = sequence.GetEffectsByShape(shape);
//Get the second animation effect of the shape
IEffect effect = shapeAnimationEffects[1];
//Remove the animation effect from the sequence
sequence.Remove(effect);
//Insert the removed animation effect as first
sequence.Insert(0, effect);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Animation.ppt
x", "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Animation.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Creating custom path animation effect

Presentation library allows you to create and modify the custom animations in presentations. The following example demonstrates how to apply a custom animation to a shape.

C#

```

//Creates an instance for PowerPoint
using (IPresentation pptxDoc = Presentation.Create())
{
    //Adds a blank slide to Presentation
    ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
    //Adds normal shape to slide
    IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 200, 0, 300,
    300);
    //Access the animation sequence to create effects
    ISequence sequence = slide.Timeline.MainSequence;
    //Add user path effect to the shape
    IEfffect bounceEffect = sequence.AddEffect(cubeShape, EffectType.PathUser,
    EffectSubtype.None, EffectTriggerType.OnClick);
    //Add commands to the empty path for moving
    IMotionEffect motionBehavior = ((IMotionEffect)bounceEffect.Behaviors[0]);
    PointF[] points = new PointF[1];
    //Add the move command to move the position of the shape
    points[0] = new PointF(0, 0);
    motionBehavior.Path.Add(MotionCommandPathType.MoveTo, points,
    MotionPathPointsType.Auto, false);
    //Add the line command to move the shape in straight line
    points[0] = new PointF(0, 0.25f);
    motionBehavior.Path.Add(MotionCommandPathType.LineTo, points,
    MotionPathPointsType.Auto, false);
    //Add the end command to finish the path animation
    motionBehavior.Path.Add(MotionCommandPathType.End, null,
    MotionPathPointsType.Auto, false);
    //Saves the Presentation
    pptxDoc.Save("Sample.pptx");
}

```

VB.NET

```

'Creates an instance for PowerPoint
Using pptxDoc As IPresentation = Presentation.Create()
'Adds a blank slide to Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds normal shape to slide
Dim cubeShape As IShape = slide.Shapes.AddShape(AutoShapeType.Cube, 200, 0,
300, 300)
'Access the animation sequence to create effects
Dim sequence As ISequence = slide.Timeline.MainSequence
'Add user path effect to the shape
Dim bounceEffect As IEfffect = sequence.AddEffect(cubeShape,
EffectType.PathUser, EffectSubtype.None, EffectTriggerType.OnClick)
'Add commands to the empty path for moving
Dim motionBehavior As IMotionEffect = DirectCast(bounceEffect.Behaviors(0),
IMotionEffect)
Dim points As PointF() = New PointF(0) {}
'Add the move command to move the position of the shape
points(0) = New PointF(0, 0)
motionBehavior.Path.Add(MotionCommandPathType.MoveTo, points,
MotionPathPointsType.Auto, False)
'Add the line command to move the shape in straight line
points(0) = New PointF(0, 0.25F)

```

```

motionBehavior.Path.Add(MotionCommandPathType.LineTo, points,
MotionPathPointsType.Auto, False)
'Add the end command to finish the path animation
motionBehavior.Path.Add(MotionCommandPathType.[End], Nothing,
MotionPathPointsType.Auto, False)
'Saves the Presentation
pptxDoc.Save("Sample.pptx")
End Using

```

UWP

```

//Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds normal shape to slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 200, 0, 300,
300);
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add user path effect to the shape
IEffect bounceEffect = sequence.AddEffect(cubeShape, EffectType.PathUser,
EffectSubtype.None, EffectTriggerType.OnClick);
//Add commands to the empty path for moving
IMotionEffect motionBehavior = ((IMotionEffect)bounceEffect.Behaviors[0]);
PointF[] points = new PointF[1];
//Add the move command to move the position of the shape
points[0] = new PointF(0, 0);
motionBehavior.Path.Add(MotionCommandPathType.MoveTo, points,
MotionPathPointsType.Auto, false);
//Add the line command to move the shape in straight line
points[0] = new PointF(0, 0.25f);
motionBehavior.Path.Add(MotionCommandPathType.LineTo, points,
MotionPathPointsType.Auto, false);
//Add the end command to finish the path animation
motionBehavior.Path.Add(MotionCommandPathType.End, null,
MotionPathPointsType.Auto, false);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds normal shape to slide

```

```

IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 200, 0, 300,
300);
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add user path effect to the shape
IEffect bounceEffect = sequence.AddEffect(cubeShape, EffectType.PathUser,
EffectSubtype.None, EffectTriggerType.OnClick);
//Add commands to the empty path for moving
IMotionEffect motionBehavior = ((IMotionEffect)bounceEffect.Behaviors[0]);
PointF[] points = new PointF[1];
//Add the move command to move the position of the shape
points[0] = new PointF(0, 0);
motionBehavior.Path.Add(MotionCommandPathType.MoveTo, points,
MotionPathPointsType.Auto, false);
//Add the line command to move the shape in straight line
points[0] = new PointF(0, 0.25f);
motionBehavior.Path.Add(MotionCommandPathType.LineTo, points,
MotionPathPointsType.Auto, false);
//Add the end command to finish the path animation
motionBehavior.Path.Add(MotionCommandPathType.End, null,
MotionPathPointsType.Auto, false);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);

```

XAMARIN

```

//Creates an instance for PowerPoint
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide to Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds normal shape to slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 200, 0, 300,
300);
//Access the animation sequence to create effects
ISequence sequence = slide.Timeline.MainSequence;
//Add user path effect to the shape
IEffect bounceEffect = sequence.AddEffect(cubeShape, EffectType.PathUser,
EffectSubtype.None, EffectTriggerType.OnClick);
//Add commands to the empty path for moving
IMotionEffect motionBehavior = ((IMotionEffect)bounceEffect.Behaviors[0]);
PointF[] points = new PointF[1];
//Add the move command to move the position of the shape
points[0] = new PointF(0, 0);
motionBehavior.Path.Add(MotionCommandPathType.MoveTo, points,
MotionPathPointsType.Auto, false);
//Add the line command to move the shape in straight line
points[0] = new PointF(0, 0.25f);
motionBehavior.Path.Add(MotionCommandPathType.LineTo, points,
MotionPathPointsType.Auto, false);
//Add the end command to finish the path animation
motionBehavior.Path.Add(MotionCommandPathType.End, null,
MotionPathPointsType.Auto, false);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.

```



```
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Removing animation effect

Presentation library allows you to remove the animation effects from a shape. The following example demonstrates how to remove an animation effect from a shape.

C#

```
//Open the existing presentation
using (IPresentation pptxDoc = Presentation.Open("Sample.pptx"))
{
//Iterate the slide
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape
IShape shape = slide.Shapes[0] as IShape;
//Iterate the sequence
ISequence sequence = slide.Timeline.MainSequence;
//To Remove the animation effects from the shape
//Get the animation effects of the particular shape
IEffect[] animationEffects = sequence.GetEffectsByShape(shape);
//Remove the animation effect from the main sequence
foreach (IEffect effect in animationEffects)
{
sequence.Remove(effect);
}
//Save the created presentation
pptxDoc.Save("Result.pptx");
}
```

VB.NET

```
'Open the existing presentation
Using pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Iterate the slide
Dim slide As ISlide = pptxDoc.Slides(0)
'Retrieves the first shape
Dim shape As IShape = TryCast(slide.Shapes(0), IShape)
'Iterate the sequence
Dim sequence As ISequence = slide.Timeline.MainSequence
'To Remove the animation effects from the shape
```

```

'Get the animation effects of the particular shape
Dim animationEffects As IEffekt() = sequence.GetEffectsByShape(shape)
'Remove the animation effect from the main sequence
For Each effect As IEffekt In animationEffects
sequence.Remove(effect)
Next
'Save the created presentation
pptxDoc.Save("Result.pptx")
End Using

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Iterate the slide
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape
IShape shape = slide.Shapes[0] as IShape;
//Iterate the sequence
ISequence sequence = slide.Timeline.MainSequence;
//To Remove the animation effects from the shape
//Get the animation effects of the particular shape
IEffekt[] animationEffects = sequence.GetEffectsByShape(shape);
//Remove the animation effect from the main sequence
foreach (IEffekt effect in animationEffects)
{
sequence.Remove(effect);
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Iterate the slide
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape
IShape shape = slide.Shapes[0] as IShape;
//Iterate the sequence
ISequence sequence = slide.Timeline.MainSequence;

```

```
//To Remove the animation effects from the shape
//Get the animation effects of the particular shape
IEffect[] animationEffects = sequence.GetEffectsByShape(shape);
//Remove the animation effect from the main sequence
foreach (IEffect effect in animationEffects)
{
    sequence.Remove(effect);
}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Animation.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Iterate the slide
ISlide slide = pptxDoc.Slides[0];
//Retrieves the first shape
IShape shape = slide.Shapes[0] as IShape;
//Iterate the sequence
ISequence sequence = slide.Timeline.MainSequence;
//To Remove the animation effects from the shape
//Get the animation effects of the particular shape
IEffect[] animationEffects = sequence.GetEffectsByShape(shape);
//Remove the animation effect from the main sequence
foreach (IEffect effect in animationEffects)
{
    sequence.Remove(effect);
}
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Animation.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Animation.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
```

Supported animation effects type

Syncfusion Presentation library supports the following predefined animation effects with the sub types like Microsoft PowerPoint.

Effects	Entrance	Exit	Emphasis	Motion path	Effect options
Appear	Yes	Yes	-	-	None
CurveUpDown	Yes	Yes	-	-	None
Ascend	Yes	Yes	-	-	None
Blast	-	-	Yes	-	None
Blinds	Yes	Yes	-	-	<ul style="list-style-type: none"> • Horizontal • Vertical
Blink	-	-	Yes	-	None
BoldFlash	-	-	Yes	-	None
BoldReveal	-	-	Yes	-	None
Boomerang	Yes	Yes	-	-	None
Bounce	Yes	Yes	-	-	None
Box	Yes	Yes	-	-	<ul style="list-style-type: none"> • In • Out
BrushOnColor	-	-	Yes	-	None
BrushOnUnderline	-	-	Yes	-	None
CenterRevolve	Yes	Yes	-	-	None
ChangeFillColor	-	-	Yes	-	<ul style="list-style-type: none"> • Instant • Gradual • GradualAndCycleClockwise • GradualAndCycleCounterClockwise
ChangeFont	-	-	Yes	-	<ul style="list-style-type: none"> • Instant • Gradual
ChangeFontColor	-	-	Yes	-	<ul style="list-style-type: none"> • Instant • Gradual • GradualAndCycleClockwise

					<ul style="list-style-type: none"> • GradualAndCycleCounterClockwise
ChangeFontSize	-	-	Yes	-	<ul style="list-style-type: none"> • Instant • Gradual
ChangeFontStyle	-	-	Yes	-	<ul style="list-style-type: none"> • FontBold • FontItalic • FontUnderline
ChangeLineColor	-	-	Yes	-	<ul style="list-style-type: none"> • Instant • Gradual • GradualAndCycleClockwise • GradualAndCycleCounterClockwise
Checkerboard	Yes	Yes	-	-	<ul style="list-style-type: none"> • Vertical • Across
Circle	Yes	Yes	-	-	<ul style="list-style-type: none"> • In • Out
ColorBlend	-	-	Yes	-	None
ColorTypewriter	Yes	Yes	-	-	None
ColorWave	-	-	Yes	-	None
ComplementaryColor	-	-	Yes	-	None
ComplementaryColor2	-	-	Yes	-	None
Compress	Yes	Yes	-	-	None
ContrastingColor	-	-	Yes	-	None
Crawl	Yes	Yes	-	-	<ul style="list-style-type: none"> • Right • Left • Top • Bottom
Credits	Yes	Yes	-	-	None
Custom	-	-	-	-	-
Darken	-	-	Yes	-	None
Desaturate	-	-	Yes	-	None

Descend	Yes	Yes	-	-	None
Diamond	Yes	Yes	-	-	<ul style="list-style-type: none"> • In • Out
Dissolve	Yes	Yes	-	-	None
EaseInOut	Yes	Yes	-	-	None
Expand	Yes	Yes	-	-	None
Fade	Yes	Yes	-	-	None
FadedSwivel	Yes	Yes	-	-	None
FadedZoom	Yes	Yes	-	-	<ul style="list-style-type: none"> • None • Center
FlashBulb	-	-	Yes	-	None
FlashOnce	Yes	Yes	-	-	None
Flicker	-	-	Yes	-	None
Flip	Yes	Yes	-	-	None
Float	Yes	Yes	-	-	None
Fly	Yes	Yes	-	-	<ul style="list-style-type: none"> • Right • Left • Top • Bottom • TopLeft • TopRight • BottomLeft • BottomRight
Fold	Yes	Yes	-	-	None
Glide	Yes	Yes	-	-	None
GrowAndTurn	Yes	Yes	-	-	None
GrowShrink	-	-	Yes	-	None
GrowWithColor	-	-	Yes	-	None
Lighten	-	-	Yes	-	None
LightSpeed	Yes	Yes	-	-	None

Path4PointStar	-	-	-	Yes	None
Path5PointStar	-	-	-	Yes	None
Path6PointStar	-	-	-	Yes	None
Path8PointStar	-	-	-	Yes	None
PathArcDown	-	-	-	Yes	None
PathArcLeft	-	-	-	Yes	None
PathArcRight	-	-	-	Yes	None
PathArcUp	-	-	-	Yes	None
PathBean	-	-	-	Yes	None
PathBounceLeft	-	-	-	Yes	None
PathBounceRight	-	-	-	Yes	None
PathBuzzsaw	-	-	-	Yes	None
PathCircle	-	-	-	Yes	None
PathCrescentMoon	-	-	-	Yes	None
PathCurvedSquare	-	-	-	Yes	None
PathCurvedX	-	-	-	Yes	None
PathCurvyLeft	-	-	-	Yes	None
PathCurvyRight	-	-	-	Yes	None
PathCurvyStar	-	-	-	Yes	None
PathDecayingWave	-	-	-	Yes	None
PathDiagonalDownRight	-	-	-	Yes	None
PathDiagonalUpRight	-	-	-	Yes	None
PathDiamond	-	-	-	Yes	None
PathDown	-	-	-	Yes	None
PathEqualTriangle	-	-	-	Yes	None
PathFigure8Four	-	-	-	Yes	None
PathFootball	-	-	-	Yes	None
PathFunnel	-	-	-	Yes	None
PathHeart	-	-	-	Yes	None

PathHeartbeat	-	-	-	Yes	None
PathHexagon	-	-	-	Yes	None
PathHorizontalFigure8	-	-	-	Yes	None
PathInvertedSquare	-	-	-	Yes	None
PathInvertedTriangle	-	-	-	Yes	None
PathLeft	-	-	-	Yes	None
PathLoopdeLoop	-	-	-	Yes	None
PathNeutron	-	-	-	Yes	None
PathOctagon	-	-	-	Yes	None
PathParallelogram	-	-	-	Yes	None
PathPeanut	-	-	-	Yes	None
PathPentagon	-	-	-	Yes	None
PathPlus	-	-	-	Yes	None
PathPointyStar	-	-	-	Yes	None
PathRight	-	-	-	Yes	None
PathRightTriangle	-	-	-	Yes	None
PathSCurve1	-	-	-	Yes	None
PathSCurve2	-	-	-	Yes	None
PathSineWave	-	-	-	Yes	None
PathSpiralLeft	-	-	-	Yes	None
PathSpiralRight	-	-	-	Yes	None
PathSpring	-	-	-	Yes	None
PathSquare	-	-	-	Yes	None
PathStairsDown	-	-	-	Yes	None
PathSwoosh	-	-	-	Yes	None
PathTeardrop	-	-	-	Yes	None
PathTrapezoid	-	-	-	Yes	None
PathTurnDown	-	-	-	Yes	None
PathTurnRight	-	-	-	Yes	None

PathTurnUp	-	-	-	Yes	None
PathTurnUpRight	-	-	-	Yes	None
PathUp	-	-	-	Yes	None
PathUser	-	-	-	Yes	None
PathVerticalFigure8	-	-	-	Yes	None
PathWave	-	-	-	Yes	None
PathZigzag	-	-	-	Yes	None
Peek	Yes	Yes	-	-	<ul style="list-style-type: none"> • Bottom • Left • Right • Top
Pinwheel	Yes	Yes	-	-	None
Plus	Yes	Yes	-	-	<ul style="list-style-type: none"> • In • Out
RandomBars	Yes	Yes	-	-	<ul style="list-style-type: none"> • Horizontal • Vertical
RandomEffects	Yes	Yes	-	-	None
RiseUp	Yes	-	-	-	None
Shimmer	-	-	Yes	-	None
Sling	Yes	Yes	-	-	None
Spin	-	-	Yes	-	None
Spinner	-	-	Yes	-	None
Spiral	Yes	Yes	-	-	None
Split	Yes	Yes	-	-	<ul style="list-style-type: none"> • HorizontalIn • HorizontalOut • VerticalIn • VerticalOut
Stretch	yes	Yes	-	-	<ul style="list-style-type: none"> • Right • Left • Top

					<ul style="list-style-type: none"> • Bottom • Across
Strips	Yes	Yes	-	-	<ul style="list-style-type: none"> • UpLeft • UpRight • DownLeft • DownRight
StyleEmphasis	-	-	Yes	-	None
Swish	Yes	Yes	-	-	None
Swivel	Yes	Yes	-	-	<ul style="list-style-type: none"> • Horizontal • Vertical
Teeter	-	-	Yes	-	None
Thread	-	-	Yes	-	None
Transparency	-	-	Yes	-	None
Unfold	yes	Yes	-	-	None
VerticalGrow	-	-	Yes	-	None
Wave	-	-	Yes	-	None
Wedge	Yes	Yes	-	-	None
Wheel	Yes	Yes	-	-	<ul style="list-style-type: none"> • Wheel1 • Wheel2 • Wheel3 • Wheel4 • Wheel8
Whip	yes	Yes	-	-	None
Wipe	Yes	Yes	-	-	<ul style="list-style-type: none"> • Top • Right • Bottom • Left
Magnify	Yes	Yes	-	-	None
Zoom	Yes	Yes	-	-	<ul style="list-style-type: none"> • In • Out

					<ul style="list-style-type: none"> • InCenter - only for Entrance type • OutBottom - only for Entrance type • OutSlightly • InSlightly • OutCenter - only for Exit type • InBottom - only for Exit type
--	--	--	--	--	---

Add and edit transitions in PowerPoint slides

Slide transitions are the motion effects that occur when you move from one slide to the next during a slide show presentation. A transition can be simple as push type that pushes to the next slide or an airplane type that displays the next slide like an eye-catching effect. You can control the speed, add sound, and customize the properties of transition effects. Transition effects can be grouped into three categories.

1. Subtle
2. Exciting
3. Dynamic Content

Transition effect contains the following properties. This makes slide transition more flexible and interactive.

1. Start (after a click or after certain time)
2. Duration
3. Speed

Set a transition effect to a PowerPoint slide

The following code example demonstrates how to set a transition effect to a PowerPoint slide.

C#

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Set the transition effect options
slide.SlideTransition.TransitionEffectOption = TransitionEffectOption.Across;
//Save the presentation
pptxDoc.Save("Sample.pptx");
//Close the presentation
pptxDoc.Close();
```

VB.NET

```

'Create a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Add a blank slide to the presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a shape to the slide
Dim cubeShape As IShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200,
300, 300)
'Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard
'Set the transition effect option
slide.SlideTransition.TransitionEffectOption = TransitionEffectOption.Across
'Save the presentation
pptxDoc.Save("Sample.pptx")
'Close the presentation
pptxDoc.Close()

```

UWP

```

//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Set the transition effect options
slide.SlideTransition.TransitionEffectOption =
TransitionEffectOption.Across;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;

```

```
//Set the transition effect options
slide.SlideTransition.TransitionEffectOption =
TransitionEffectOption.Across;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();
```

XAMARIN

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Set the transition effect options
slide.SlideTransition.TransitionEffectOption =
TransitionEffectOption.Across;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Modify a transition effect applied to a PowerPoint slide

You can edit the transition effects that already applied to the PowerPoint slides. Refer to the following code example.

C#

```
//Open an existing PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Retrieve the first slide from the presentation
ISlide slide = pptxDoc.Slides[0];
//Modify the transition effect applied to the slide
slide.SlideTransition.TransitionEffect = TransitionEffect.Cover;
```

```
//Set the transition subtype
slide.SlideTransition.TransitionEffectOption = TransitionEffectOption.Right;
//Save the presentation
pptxDoc.Save("Transition.pptx");
//Close the presentation
pptxDoc.Close();
```

VB.NET

```
'Open an existing PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Retrieve the first slide from the presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Modify the transition effect applied to the slide
slide.SlideTransition.TransitionEffect = TransitionEffect.Cover
'Set the transition subtype
slide.SlideTransition.TransitionEffectOption = TransitionEffectOption.Right
'Save the presentation
pptxDoc.Save("Transition.pptx")
'Close the presentation
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Retrieve the first slide from the presentation
ISlide slide = pptxDoc.Slides[0];
//Modify the transition effect applied to the slide
slide.SlideTransition.TransitionEffect = TransitionEffect.Cover;
//Set the transition subtype
slide.SlideTransition.TransitionEffectOption = TransitionEffectOption.Right;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Transition";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieve the first slide from the presentation
```

```
ISlide slide = pptxDoc.Slides[0];
//Modify the transition effect applied to the slide
slide.SlideTransition.TransitionEffect = TransitionEffect.Cover;
//Set the transition subtype
slide.SlideTransition.TransitionEffectOption = TransitionEffectOption.Right;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Transition.pptx",
    FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream("Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Retrieve the first slide from the presentation
ISlide slide = pptxDoc.Slides[0];
//Modify the transition effect applied to the slide
slide.SlideTransition.TransitionEffect = TransitionEffect.Cover;
//Set the transition subtype
slide.SlideTransition.TransitionEffectOption = TransitionEffectOption.Right;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Transition.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Transition.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);
```

Set the transition duration

You can set the transition duration value up to 59 seconds. This specifies the length of the slide transition to happen. Refer to the following code example.

C#

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
```

```
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
// Set the duration in seconds for the transition effect. Maximum duration
value is 59 seconds
slide.SlideTransition.Duration = 40;
//Save the presentation
pptxDoc.Save("Transition.pptx");
//Close the presentation
pptxDoc.Close();
```

VB.NET

```
'Create a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Add a blank slide to the presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a shape to the slide
Dim cubeShape As IShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200,
300, 300)
'Add a shape to the slide
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard
'Set the duration value(sec) for the transition effect. Max duration value
is 59 seconds
slide.SlideTransition.Duration = 40
'Save the presentation
pptxDoc.Save("Transition.pptx")
'Close the presentation
pptxDoc.Close()
```

UWP

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
// Set the duration in seconds for the transition effect. Maximum duration
value is 59 seconds
slide.SlideTransition.Duration = 40;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Transition";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```


ASP.NET CORE

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
// Set the duration in seconds for the transition effect. Maximum duration value is 59 seconds
slide.SlideTransition.Duration = 40;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Transition.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();
```

XAMARIN

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
// Set the duration in seconds for the transition effect. Maximum duration value is 59 seconds
slide.SlideTransition.Duration = 40;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Transition.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Transition.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
```

Set the transition delay

You can set the transition delay in seconds. This delays the next transactions to happen for a certain number of seconds. The following example demonstrates how to apply the time delay.

C#

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Enable the transition time delay
slide.SlideTransition.TriggerOnTimeDelay = true;
//Assign the value for the advance time delay in seconds
slide.SlideTransition.TimeDelay = 5;
//Save the presentation
pptxDoc.Save("Sample.pptx");
//Close the presentation
pptxDoc.Close();
```

VB.NET

```
'Create a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Add a blank slide to the presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a shape to the slide
Dim cubeShape As IShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300)
'Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard
'Enable the transition time delay
slide.SlideTransition.TriggerOnTimeDelay = True
'Assign the value for the advance time delay in seconds
slide.SlideTransition.TimeDelay = 5
'Save the presentation
pptxDoc.Save("Sample.pptx")
'Close the presentation
pptxDoc.Close()
```

UWP

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300, 300);
//Set the transition effect type
```

```

slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Enable the transition time delay
slide.SlideTransition.TriggerOnTimeDelay = true;
//Assign the value for the advance time delay in seconds
slide.SlideTransition.TimeDelay = 5;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Enable the transition time delay
slide.SlideTransition.TriggerOnTimeDelay = true;
//Assign the value for the advance time delay in seconds
slide.SlideTransition.TimeDelay = 5;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();

```

XAMARIN

```

//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Enable the transition time delay
slide.SlideTransition.TriggerOnTimeDelay = true;
//Assign the value for the advance time delay in seconds
slide.SlideTransition.TimeDelay = 5;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);

```

```
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Set the trigger mode for the transition

The next slide transition can be triggered by the following two ways:

1. Mouse clicks - Brings the next slide to the view.
2. Setting a time - Brings the next slide after that specified time without any interactions.

Syncfusion PowerPoint library allows you to set both the previously given trigger modes while using PowerPoint slide transitions. Refer to the following code example.

C#

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Set transition advance on click to true. This will enable the next
transition after a click
slide.SlideTransition.TriggerOnClick = true;
//Save the presentation
pptxDoc.Save("Sample.pptx");
//Close the presentation
pptxDoc.Close();
```

VB.NET

```
'Create a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Add a blank slide to the presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a shape to the slide
Dim cubeShape As IShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200,
300, 300)
'Set the transition effect type
```

```

slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard
'Set transition advance on click to true. This will enable the next
transition after a click
slide.SlideTransition.TriggerOnClick = True
'Save the presentation
pptxDoc.Save("Sample.pptx")
'Close the presentation
pptxDoc.Close()

```

UWP

```

//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Set transition advance on click to true. This will enable the next
transition after a click
slide.SlideTransition.TriggerOnClick = true;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Set transition advance on click to true. This will enable the next
transition after a click
slide.SlideTransition.TriggerOnClick = true;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();

```

XAMARIN

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Set transition advance on click to true. This will enable the next
transition after a click
slide.SlideTransition.TriggerOnClick = true;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Set the speed for transition effect

The speed is the customized property provided by Syncfusion PowerPoint library to set the transition duration mentioned [above](#) (in this page) to a customized enumeration values. By default, each transition will happen for 2 seconds. You can change the following enumeration values to change the duration of a slide transition:

1. Default - 2 seconds
2. Fast - 0.5 seconds
3. Slow - 1.0 seconds
4. Medium - 0.75 seconds

C#

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
```

```
//Set the transition effect speed enumeration. This will reduce the
transition duration to 0.75 seconds from the default 2 second
slide.SlideTransition.Speed = TransitionSpeed.Medium;
//Save the presentation
pptxDoc.Save("Sample.pptx");
```

VB.NET

```
'Create a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Add a blank slide to the presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a shape to the slide
Dim cubeShape As IShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200,
300, 300)
'Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard
'Set the transition effect speed enumeration. This will reduce the
transition duration to 0.75 seconds from the default 2 seconds
slide.SlideTransition.Speed = TransitionSpeed.Medium
'Save the presentation
pptxDoc.Save("Sample.pptx")
```

UWP

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Set the transition effect speed enumeration. This will reduce the
transition duration to 0.75 seconds from the default 2 second
slide.SlideTransition.Speed = TransitionSpeed.Medium;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
```

```

IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Set the transition effect speed enumeration. This will reduce the
transition duration to 0.75 seconds from the default 2 second
slide.SlideTransition.Speed = TransitionSpeed.Medium;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);

```

XAMARIN

```

//Create a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a shape to the slide
IShape cubeShape = slide.Shapes.AddShape(AutoShapeType.Cube, 50, 200, 300,
300);
//Set the transition effect type
slide.SlideTransition.TransitionEffect = TransitionEffect.Checkerboard;
//Set the transition effect speed enumeration. This will reduce the
transition duration to 0.75 seconds from the default 2 second
slide.SlideTransition.Speed = TransitionSpeed.Medium;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Supported transition effect types:

The following are the list of transition effect options that can be applied to each transition effects.

S.No	Effects	Effect Options
1	Airplane	<ul style="list-style-type: none"> Left Right
2	Blinds	<ul style="list-style-type: none"> Horizontal

		<ul style="list-style-type: none"> • Vertical
3	Box	<ul style="list-style-type: none"> • Left • Right • Up • Down
4	Checkerboard	<ul style="list-style-type: none"> • Across • Down
5	Circle	None
6	Comb	<ul style="list-style-type: none"> • Horizontal • Vertical
7	Conveyor	<ul style="list-style-type: none"> • Left • Right
8	Cover	<ul style="list-style-type: none"> • Left • Right • Up • Down • LeftUp • LeftDown • RightUp • RightDown
9	Crush	None
10	Cube	<ul style="list-style-type: none"> • Left • Right • Up • Down
11	Curtains	None
12	Cut	<ul style="list-style-type: none"> • None • ThroughBlack
13	Diamond	None
14	Dissolve	None

15	Doors	<ul style="list-style-type: none"> • Horizontal • Vertical
16	Drape	<ul style="list-style-type: none"> • Left • Right
17	FadeAway	<ul style="list-style-type: none"> • Smoothly • ThroughBlack
18	FallOver	<ul style="list-style-type: none"> • Left • Right
19	FerrisWheel	<ul style="list-style-type: none"> • Left • Right
20	Flashbulb	None
21	Flip	<ul style="list-style-type: none"> • Left • Right
22	FlyThrough	<ul style="list-style-type: none"> • In • Out • InBounce • OutBounce
23	Fracture	None
24	Gallery	<ul style="list-style-type: none"> • Left • Right
25	GlitterDiamond	<ul style="list-style-type: none"> • Left • Right • Up • Down
26	GlitterHexagon	<ul style="list-style-type: none"> • Left • Right • Up • Down
27	Honeycomb	None
28	Morph	<ul style="list-style-type: none"> • ByObject

		<ul style="list-style-type: none"> • ByWord ByChar
29	Newsflash	None
30	Orbit	<ul style="list-style-type: none"> • Left • Right • Up • Down
31	Origami	<ul style="list-style-type: none"> • Left • Right
32	PageCurlDouble	<ul style="list-style-type: none"> • Left • Right
33	PageCurlSingle	<ul style="list-style-type: none"> • Left • Right
34	Pan	<ul style="list-style-type: none"> • Left • Right • Up • Down
35	PeelOff	<ul style="list-style-type: none"> • Left • Right
36	Plus	None
37	Prestige	None
38	Push	<ul style="list-style-type: none"> • Left • Right • Up • Down
39	Random	None
40	RandomBars	<ul style="list-style-type: none"> • Horizontal • Vertical
41	Reveal	<ul style="list-style-type: none"> • SmoothLeft • SmoothRight • BlackLeft

		<ul style="list-style-type: none"> • BlackRight
42	Ripple	<ul style="list-style-type: none"> • Center • RightUp • RightDown • LeftUp • LeftDown
43	Rotate	<ul style="list-style-type: none"> • Left • Right • Up • Down
44	Shred	<ul style="list-style-type: none"> • StripsIn • StripsOut • RectangleIn • RectangleOut
45	Split	<ul style="list-style-type: none"> • HorizontalIn • HorizontalOut • VerticalIn • VerticalOut
46	Strips	<ul style="list-style-type: none"> • LeftDown • LeftUp • RightDown • RightUp
47	Switch	<ul style="list-style-type: none"> • Left • Right
48	Uncover	<ul style="list-style-type: none"> • Left • Right • Up • Down • LeftDown • LeftUp • RightDown

		<ul style="list-style-type: none"> • RightUp
49	Vortex	<ul style="list-style-type: none"> • Left • Right • Up • Down
50	Warp	<ul style="list-style-type: none"> • In • Out
51	Wedge	None
52	Wheel	<ul style="list-style-type: none"> • Spoke1 • Spoke2 • Spoke3 • Spoke4 • Spoke8 • Reverse1Spoke
53	Wind	<ul style="list-style-type: none"> • Left • Right
54	Window	<ul style="list-style-type: none"> • Horizontal • Vertical
55	Wipe	<ul style="list-style-type: none"> • Left • Right • Up • Down
56	Zoom	<ul style="list-style-type: none"> • In • Out

Adding connectors in PowerPoint slides

Essential Presentation library supports adding, editing, and removing the connectors in a PowerPoint file. The following code example demonstrates how to add a connector between two shapes.

The following code example demonstrates how to add a connector in PowerPoint slide.

C#

```
//Create a new PowerPoint file
using (IPresentation pptxDoc = Presentation.Create())
{
```

```
//Add a slide to the PowerPoint file
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a rectangle shape on the slide
IShape rectangle = slide.Shapes.AddShape(AutoShapeType.Rectangle, 200, 300,
100, 100);
//Add an oval shape on the slide
IShape oval = slide.Shapes.AddShape(AutoShapeType.Oval, 400, 10, 100, 100);
//Add elbow connector on the slide and connect the end points of connector
with specified port positions 0 and 4 of the beginning and end shapes
IConnector connector = slide.Shapes.AddConnector(ConnectorType.Elbow,
rectangle, 0, oval, 4);
//Save the PowerPoint file
pptxDoc.Save("Sample.pptx");
}
```

VB.NET

```
'Create a new PowerPoint file
Using pptxDoc As IPresentation = Presentation.Create
'Add a slide to the PowerPoint file
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a rectangle shape on the slide
Dim rectangle As IShape = slide.Shapes.AddShape(AutoShapeType.Rectangle,
200, 300, 100, 100)
'Add an oval shape on the slide
Dim oval As IShape = slide.Shapes.AddShape(AutoShapeType.Oval, 400, 10, 100,
100)
'Add elbow connector on the slide and connect the end points of connector
with specified port positions 0 and 4 of the beginning and end shapes
Dim connector As IConnector = slide.Shapes.AddConnector(ConnectorType.Elbow,
rectangle, 0, oval, 4)
'Save the PowerPoint file
pptxDoc.Save("Sample.pptx")
End Using
```

UWP

```
//Create a new PowerPoint file
using (IPresentation pptxDoc = Presentation.Create())
{
//Add a slide to the PowerPoint file
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a rectangle shape on the slide
IShape rectangle = slide.Shapes.AddShape(AutoShapeType.Rectangle, 200, 300,
100, 100);
//Add an oval shape on the slide
IShape oval = slide.Shapes.AddShape(AutoShapeType.Oval, 400, 10, 100, 100);
//Add elbow connector on the slide and connect the end points of connector
with specified port positions 0 and 4 of the beginning and end shapes
IConnector connector = slide.Shapes.AddConnector(ConnectorType.Elbow,
rectangle, 0, oval, 4);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
}
```

```

savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
}

```

ASP.NET CORE

```

using (IPresentation pptxDoc = Presentation.Create())
{
    //Add a slide to the PowerPoint file
    ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
    //Add a rectangle shape on the slide
    IShape rectangle = slide.Shapes.AddShape(AutoShapeType.Rectangle, 200, 300,
    100, 100);
    //Add an oval shape on the slide
    IShape oval = slide.Shapes.AddShape(AutoShapeType.Oval, 400, 10, 100, 100);
    //Add elbow connector on the slide and connect the end points of connector
    with specified port positions 0 and 4 of the beginning and end shapes
    IConnector connector = slide.Shapes.AddConnector(ConnectorType.Elbow,
    rectangle, 0, oval, 4);
    //Save the PowerPoint Presentation as stream
    FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
    pptxDoc.Save(outputStream);
}

```

XAMARIN

```

//Create a new PowerPoint file
using (IPresentation pptxDoc = Presentation.Create())
{
    //Add a slide to the PowerPoint file
    ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
    //Add a rectangle shape on the slide
    IShape rectangle = slide.Shapes.AddShape(AutoShapeType.Rectangle, 200, 300,
    100, 100);
    //Add an oval shape on the slide
    IShape oval = slide.Shapes.AddShape(AutoShapeType.Oval, 400, 10, 100, 100);
    //Add elbow connector on the slide and connect the end points of connector
    with specified port positions 0 and 4 of the beginning and end shapes
    IConnector connector = slide.Shapes.AddConnector(ConnectorType.Elbow,
    rectangle, 0, oval, 4);
    //Create new memory stream to save Presentation.
    MemoryStream stream = new MemoryStream();
    //Save Presentation in stream format.
    pptxDoc.Save(stream);
    stream.Position = 0;
    //The operation in Save under Xamarin varies between Windows Phone, Android
    and iOS platforms. Please refer presentation/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
}
```

Adding a single point connector

You can also add a connector to a source shape without any destination shapes. The following code example demonstrates how to add a connector with single point connection.

C#

```
//Create a new PowerPoint file
using (IPresentation pptxDoc = Presentation.Create())
{
    // Add a slide to the PowerPoint file.
    ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
    //Add a rectangle shape on the slide
    IShape rectangle = slide.Shapes.AddShape(AutoShapeType.Rectangle, 420, 250,
100, 100);
    //Add connector with specified bounds
    IConnector connector = slide.Shapes.AddConnector(ConnectorType.Straight, 0,
0, 470, 150);
    //Connect the beginning point of the connector with rectangle shape
    connector.BeginConnect(rectangle, 0);
    //Set the beginning cap of the connector as arrow
    connector.LineFormat.BeginArrowheadStyle = ArrowheadStyle.Arrow;
    //Change the connector color
    //Set the connector fill type as solid
    connector.LineFormat.Fill.FillType = FillType.Solid;
    //Set the connector solid fill as black
    connector.LineFormat.Fill.SolidFill.Color = ColorObject.Black;
    //Save the PowerPoint file
    pptxDoc.Save("Sample.pptx");
}
```

VB.NET

```
'Create a new PowerPoint file
Using pptxDoc As IPresentation = Presentation.Create
'Add a slide to the PowerPoint file
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a rectangle shape on the slide
Dim rectangle As IShape = slide.Shapes.AddShape(AutoShapeType.Rectangle,
420, 250, 100, 100)
'Add connector with specified bounds
Dim connector As IConnector =
slide.Shapes.AddConnector(ConnectorType.Straight, 0, 0, 470, 150)
'Connect the beginning point of the connector with rectangle shape
connector.BeginConnect(rectangle, 0)
'Set the beginning cap of the connector as arrow
connector.LineFormat.BeginArrowheadStyle = ArrowheadStyle.Arrow
```



```

'Change the connector color
'Set the connector fill type as solid
connector.LineFormat.Fill.FillType = FillType.Solid
'Set the connector solid fill as black
connector.LineFormat.Fill.SolidFill.Color = ColorObject.Black
'Save the PowerPoint file
pptxDoc.Save("Sample.pptx")
End Using

```

UWP

```

//Create a new PowerPoint file
using (IPresentation pptxDoc = Presentation.Create())
{
    // Add a slide to the PowerPoint file.
    ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
    //Add a rectangle shape on the slide
    IShape rectangle = slide.Shapes.AddShape(AutoShapeType.Rectangle, 420, 250,
    100, 100);
    //Add connector with specified bounds
    IConnector connector = slide.Shapes.AddConnector(ConnectorType.Straight, 0,
    0, 470, 150);
    //Connect the beginning point of the connector with rectangle shape
    connector.BeginConnect(rectangle, 0);
    //Set the beginning cap of the connector as arrow
    connector.LineFormat.BeginArrowheadStyle = ArrowheadStyle.Arrow;
    //Change the connector color
    //Set the connector fill type as solid
    connector.LineFormat.Fill.FillType = FillType.Solid;
    //Set the connector solid fill as black
    connector.LineFormat.Fill.SolidFill.Color = ColorObject.Black;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Sample";
    savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await pptxDoc.SaveAsync(storageFile);
}

```

ASP.NET CORE

```

//Create a new PowerPoint file
using (IPresentation pptxDoc = Presentation.Create())
{
    // Add a slide to the PowerPoint file.
    ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
    //Add a rectangle shape on the slide
    IShape rectangle = slide.Shapes.AddShape(AutoShapeType.Rectangle, 420, 250,
    100, 100);
    //Add connector with specified bounds
    IConnector connector = slide.Shapes.AddConnector(ConnectorType.Straight, 0,
    0, 470, 150);

```

```
//Connect the beginning point of the connector with rectangle shape
connector.BeginConnect(rectangle, 0);
//Set the beginning cap of the connector as arrow
connector.LineFormat.BeginArrowheadStyle = ArrowheadStyle.Arrow;
//Change the connector color
//Set the connector fill type as solid
connector.LineFormat.Fill.FillType = FillType.Solid;
//Set the connector solid fill as black
connector.LineFormat.Fill.SolidFill.Color = ColorObject.Black;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
}
```

XAMARIN

```
//Create a new PowerPoint file
using (IPresentation pptxDoc = Presentation.Create())
{
    // Add a slide to the PowerPoint file.
    ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
    //Add a rectangle shape on the slide
    IShape rectangle = slide.Shapes.AddShape(AutoShapeType.Rectangle, 420, 250,
    100, 100);
    //Add connector with specified bounds
    IConnector connector = slide.Shapes.AddConnector(ConnectorType.Straight, 0,
    0, 470, 150);
    //Connect the beginning point of the connector with rectangle shape
    connector.BeginConnect(rectangle, 0);
    //Set the beginning cap of the connector as arrow
    connector.LineFormat.BeginArrowheadStyle = ArrowheadStyle.Arrow;
    //Change the connector color
    //Set the connector fill type as solid
    connector.LineFormat.Fill.FillType = FillType.Solid;
    //Set the connector solid fill as black
    connector.LineFormat.Fill.SolidFill.Color = ColorObject.Black;
    //The operation in Save under Xamarin varies between Windows Phone, Android
    and iOS platforms. Please refer presentation/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);
    else
        Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);
}
```

Editing a connector in PowerPoint slide

The following code example demonstrates how to edit an existing connector in a PowerPoint slide.

C#

```
//Open an existing PowerPoint file
```

```

using (IPresentation pptxDoc = Presentation.Open("Sample.pptx"))
{
    //Get the first slide of a PowerPoint file
    ISlide slide = pptxDoc.Slides[0];
    //Get the connector from a slide
    IConnector connector = slide.Shapes[2] as IConnector;
    //Set the begin cap for the connector
    connector.LineFormat.BeginArrowheadStyle = ArrowheadStyle.ArrowOpen;
    //Set the line format for the connector
    connector.LineFormat.DashStyle = LineDashStyle.DashDotDot;
    //Disconnect the end connection of the connector if end point get connected
    if (connector.EndConnectedShape != null)
        connector.EndDisconnect();
    //Insert a triangle shape into slide
    IShape triangle = slide.Shapes.AddShape(AutoShapeType.IsoscelesTriangle,
    600, 500, 150, 150);
    //Declare the end connection site index
    int connectionSiteIndex = 4;
    //Reconnect the end point of connector with triangle shape if its connection
    site count is greater than 4
    if (connectionSiteIndex < triangle.ConnectionSiteCount)
        connector.EndConnect(triangle, connectionSiteIndex);
    //Save the PowerPoint file
    pptxDoc.Save("Connector.pptx");
}

```

VB.NET

```

' Open an existing PowerPoint file.
Using pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Get the first slide of a PowerPoint file
Dim slide As ISlide = pptxDoc.Slides(2)
'Get the connector from a slide
Dim connector As IConnector = CType(slide.Shapes(2), IConnector)
'Set the beginning cap for the connector
connector.LineFormat.BeginArrowheadStyle = ArrowheadStyle.ArrowOpen
'Set the line format for the connector
connector.LineFormat.DashStyle = LineDashStyle.DashDotDot
'Disconnect the end connection of the connector if end point get connected
If (Not (connector.EndConnectedShape) Is Nothing) Then
    connector.EndDisconnect()
End If
'Insert a triangle shape into slide
Dim triangle As IShape =
slide.Shapes.AddShape(AutoShapeType.IsoscelesTriangle, 600, 500, 150, 150)
'Declare the end connection site index
Dim connectionSiteIndex As Integer = 4
'Reconnect the end point of connector with triangle shape if its connection
site count is greater than 4
If (connectionSiteIndex < triangle.ConnectionSiteCount) Then
    connector.EndConnect(triangle, connectionSiteIndex)
End If
'Save the PowerPoint file
pptxDoc.Save("Connector.pptx")
End Using

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
using (IPresentation pptxDoc= await
Presentation.OpenAsync(inputStorageFile))
{
    //Get the first slide of a PowerPoint file
    ISlide slide = pptxDoc.Slides[0];
    //Get the connector from a slide
    IConnector connector = slide.Shapes[2] as IConnector;
    //Set the begin cap for the connector
    connector.LineFormat.BeginArrowheadStyle = ArrowheadStyle.ArrowOpen;
    //Set the line format for the connector
    connector.LineFormat.DashStyle = LineDashStyle.DashDotDot;
    //Disconnect the end connection of the connector if end point get connected
    if (connector.EndConnectedShape != null)
        connector.EndDisconnect();
    //Insert a triangle shape into slide
    IShape triangle = slide.Shapes.AddShape(AutoShapeType.IsoscelesTriangle,
        600, 500, 150, 150);
    //Declare the end connection site index
    int connectionSiteIndex = 4;
    //Reconnect the end point of connector with triangle shape if its connection
    site count is greater than 4
    if (connectionSiteIndex < triangle.ConnectionSiteCount)
        connector.EndConnect(triangle, connectionSiteIndex);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Connector";
    savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
        ".pptx" });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await pptxDoc.SaveAsync(storageFile);
}

```

ASP.NET CORE

```

//Loads an PowerPoint file in stream
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
//Opens the loaded PowerPoint file
using (IPresentation pptxDoc = Presentation.Open(inputStream))
{
    //Get the first slide of a PowerPoint file
    ISlide slide = pptxDoc.Slides[0];
    //Get the connector from a slide
    IConnector connector = slide.Shapes[2] as IConnector;
    //Set the begin cap for the connector

```

```

connector.LineFormat.BeginArrowheadStyle = ArrowheadStyle.ArrowOpen;
//Set the line format for the connector
connector.LineFormat.DashStyle = LineDashStyle.DashDotDot;
//Disconnect the end connection of the connector if end point get connected
if (connector.EndConnectedShape != null)
connector.EndDisconnect();
//Insert a triangle shape into slide
IShape triangle = slide.Shapes.AddShape(AutoShapeType.IsoscelesTriangle,
600, 500, 150, 150);
//Declare the end connection site index
int connectionSiteIndex = 4;
//Reconnect the end point of connector with triangle shape if its connection
site count is greater than 4
if (connectionSiteIndex < triangle.ConnectionSiteCount)
connector.EndConnect(triangle, connectionSiteIndex);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Connector.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
}

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Opens the loaded PowerPoint file
using (IPresentation pptxDoc = Presentation.Open(inputStream))
{
//Get the first slide of a PowerPoint file
ISlide slide = pptxDoc.Slides[0];
//Get the connector from a slide
IConnector connector = slide.Shapes[2] as IConnector;
//Set the begin cap for the connector
connector.LineFormat.BeginArrowheadStyle = ArrowheadStyle.ArrowOpen;
//Set the line format for the connector
connector.LineFormat.DashStyle = LineDashStyle.DashDotDot;
//Disconnect the end connection of the connector if end point get connected
if (connector.EndConnectedShape != null)
connector.EndDisconnect();
//Insert a triangle shape into slide
IShape triangle = slide.Shapes.AddShape(AutoShapeType.IsoscelesTriangle,
600, 500, 150, 150);
//Declare the end connection site index
int connectionSiteIndex = 4;
//Reconnect the end point of connector with triangle shape if its connection
site count is greater than 4
if (connectionSiteIndex < triangle.ConnectionSiteCount)
connector.EndConnect(triangle, connectionSiteIndex);
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Connector.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Connector.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
}
```

Updating the positions of the connector in PowerPoint slide

The following code example demonstrates how to update a connector's position when a source shape position is changed.

C#

```
//Open an existing PowerPoint file
using (IPresentation pptxDoc = Presentation.Open("Sample.pptx"))
{
    //Get the first slide of a PowerPoint file
    ISlide slide = pptxDoc.Slides[0];
    //Get the rectangle shape from a slide
    IShape rectangle = slide.Shapes[0] as IShape;
    //Get the connector from a slide
    IConnector connector = slide.Shapes[2] as IConnector;
    //Change the X and Y position of the rectangle
    rectangle.Left = 600;
    rectangle.Top = 200;
    //Update the connector to connect with previously updated shape
    connector.Update();
    //Save the PowerPoint file
    pptxDoc.Save("Connector.pptx");
}
```

VB.NET

```
'Open an existing PowerPoint file
Using pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Get the first slide of a PowerPoint file
Dim slide As ISlide = pptxDoc.Slides(0)
'Get the rectangle shape from a slide
Dim rectangle As IShape = CType(slide.Shapes(0), IShape)
'Get the connector from a slide
Dim connector As IConnector = CType(slide.Shapes(2), IConnector)
'Change the X and Y position of the rectangle
rectangle.Left = 600
rectangle.Top = 200
'Update the connector to connect with previously updated shape
connector.Update()
'Save the PowerPoint file
pptxDoc.Save("Connector.pptx")
End Using
```

UWP

```
//Instantiates the File Picker
```

```

FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
using (IPresentation pptxDoc= await
Presentation.OpenAsync(inputStorageFile))
{
//Get the first slide of a PowerPoint file
ISlide slide = pptxDoc.Slides[0];
//Get the rectangle shape from a slide
IShape rectangle = slide.Shapes[0] as IShape;
//Get the connector from a slide
IConnector connector = slide.Shapes[2] as IConnector;
//Change the X and Y position of the rectangle
rectangle.Left = 600;
rectangle.Top = 200;
//Update the connector to connect with previously updated shape
connector.Update();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Connector";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
}

```

ASP.NET CORE

```

//Loads an PowerPoint file in stream
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
//Opens the loaded PowerPoint file
using (IPresentation pptxDoc = Presentation.Open(inputStream))
{
//Get the first slide of a PowerPoint file
ISlide slide = pptxDoc.Slides[0];
//Get the rectangle shape from a slide
IShape rectangle = slide.Shapes[0] as IShape;
//Get the connector from a slide
IConnector connector = slide.Shapes[2] as IConnector;
//Change the X and Y position of the rectangle
rectangle.Left = 600;
rectangle.Top = 200;
//Update the connector to connect with previously updated shape
connector.Update();
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Connector.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
}

```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Opens the loaded PowerPoint file
using (IPresentation pptxDoc = Presentation.Open(inputStream))
{
    //Get the first slide of a PowerPoint file
    ISlide slide = pptxDoc.Slides[0];
    //Get the rectangle shape from a slide
    IShape rectangle = slide.Shapes[0] as IShape;
    //Get the connector from a slide
    IConnector connector = slide.Shapes[2] as IConnector;
    //Change the X and Y position of the rectangle
    rectangle.Left = 600;
    rectangle.Top = 200;
    //Update the connector to connect with previously updated shape
    connector.Update();
    //The operation in Save under Xamarin varies between Windows Phone, Android
    and iOS platforms. Please refer presentation/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Connector.pptx",
            "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
    else
        Xamarin.Forms.DependencyService.Get<ISave>().Save("Connector.pptx",
            "application/vnd.openxmlformats-officedocument.presentationml.presentation",
            stream);
}
```

Removing a connector from PowerPoint shapes

The following code example demonstrates how to remove a connector from PowerPoint slide.

C#

```
//Open an existing PowerPoint file
using (IPresentation pptxDoc = Presentation.Open("Sample.pptx"))
{
    //Get the first slide of a PowerPoint file
    ISlide slide = pptxDoc.Slides[0];
    //Get the connector from a slide
    IConnector connector = slide.Shapes[2] as IConnector;
    //Remove the connector from slide
    slide.Shapes.Remove(connector);
    //Save the PowerPoint file
    pptxDoc.Save("Connector.pptx");
}
```

VB.NET

```
'Open an existing PowerPoint file
```



```

Using pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Get the first slide of a PowerPoint file
Dim slide As ISlide = pptxDoc.Slides(0)
'Get the connector from a slide
Dim connector As IConnector = CType(slide.Shapes(2), IConnector)
'Remove the connector from slide
slide.Shapes.Remove(connector)
'Save the PowerPoint file
pptxDoc.Save("Connector.pptx")
End Using

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
using (IPresentation pptxDoc= await
Presentation.OpenAsync(inputStorageFile))
{
    //Get the first slide of a PowerPoint file
    ISlide slide = pptxDoc.Slides[0];
    //Get the connector from a slide
    IConnector connector = slide.Shapes[2] as IConnector;
    //Remove the connector from slide
    slide.Shapes.Remove(connector);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Connector";
    savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
        ".pptx" });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await pptxDoc.SaveAsync(storageFile);
}

```

ASP.NET CORE

```

//Loads an PowerPoint file in stream
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
//Opens the loaded PowerPoint file
using (IPresentation pptxDoc = Presentation.Open(inputStream))
{
    //Get the first slide of a PowerPoint file
    ISlide slide = pptxDoc.Slides[0];
    //Get the connector from a slide
    IConnector connector = slide.Shapes[2] as IConnector;
    //Remove the connector from slide
    slide.Shapes.Remove(connector);
    //Save the PowerPoint Presentation as stream
    FileStream outputStream = new FileStream("Connector.pptx", FileMode.Create);

```

```
pptxDoc.Save(outputStream);
}
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Opens the loaded PowerPoint file
using (IPresentation pptxDoc = Presentation.Open(inputStream))
{
//Get the first slide of a PowerPoint file
ISlide slide = pptxDoc.Slides[0];
//Get the connector from a slide
IConnector connector = slide.Shapes[2] as IConnector;
//Remove the connector from slide
slide.Shapes.Remove(connector);
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Connector.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Connector.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
}
```

Working with Headers and Footers

Add Headers and Footers in PowerPoint

Add Footer to a Slide in PowerPoint

Essential Presentation library facilitates adding Footer in a slide of the PowerPoint Presentation. Footers are useful in providing quick information about your document or data.

The following code example demonstrates how to add a footer to the presentation.

C#

```
//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Sets the visibility of Footer content in the slide
slide.HeadersFooters.Footer.Visible = true;
//Sets the text to be added to the Footer
slide.HeadersFooters.Footer.Text = "Footer content";
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
```

```
//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Creates an instance of Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds a blank slide
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Sets the visibility of Footer content in the slide
slide.HeadersFooters.Footer.Visible = True
'Sets the text to be added to the Footer
slide.HeadersFooters.Footer.Text = "Footer content"
'Adds textbox to the slide
Dim textboxShape As IShape = slide.AddTextBox(0, 0, 500, 500)
'Adds paragraph to the textbody of textbox
Dim paragraph As IParagraph = textboxShape.TextBody.AddParagraph()
'Adds a TextPart to the paragraph
Dim textPart As ITextPart = paragraph.AddTextPart()
'Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base."
'Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx")
'Closes the Presentation
pptxDoc.Close()
```

UWP

```
//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Sets the visibility of Footer content in the slide
slide.HeadersFooters.Footer.Visible = true;
//Sets the text to be added to the Footer
slide.HeadersFooters.Footer.Text = "Footer content";
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
```

```

//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Sets the visibility of Footer content in the slide
slide.HeadersFooters.Footer.Visible = true;
//Sets the text to be added to the Footer
slide.HeadersFooters.Footer.Text = "Footer content";
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Dispose the image stream
pictureStream.Dispose();
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

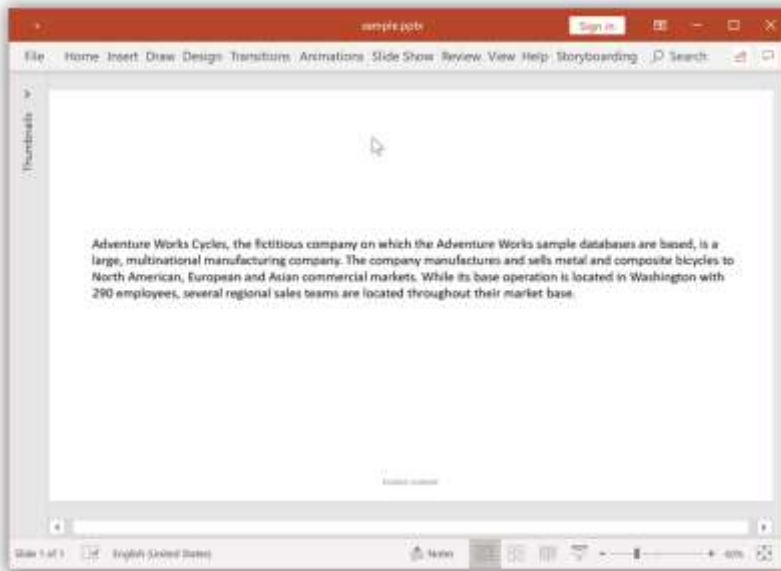
```

//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide

```

```
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);  
//Sets the visibility of Footer content in the slide  
slide.HeadersFooters.Footer.Visible = true;  
//Sets the text to be added to the Footer  
slide.HeadersFooters.Footer.Text = "Footer content";  
//Adds textbox to the slide  
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);  
//Adds paragraph to the textbody of textbox  
IParagraph paragraph = textboxShape.TextBody.AddParagraph();  
//Adds a TextPart to the paragraph  
ITextPart textPart = paragraph.AddTextPart();  
//Adds text to the TextPart  
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the  
AdventureWorks sample databases are based, is a large, multinational  
manufacturing company. The company manufactures and sells metal and  
composite bicycles to North American, European and Asian commercial markets.  
While its base operation is located in Washington with 290 employees,  
several regional sales teams are located throughout their market base.";  
//Create new memory stream to save Presentation  
MemoryStream stream = new MemoryStream();  
//Save Presentation in stream format  
pptxDoc.Save(stream);  
//Close the presentation  
pptxDoc.Close();  
stream.Position = 0;  
//The operation in Save under Xamarin varies between Windows Phone, Android  
and iOS platforms. Refer to the Presentation/Xamarin section for respective  
code samples.  
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==  
TargetPlatform.Windows)  
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",  
"application/vnd.openxmlformats-officedocument.presentationml.presentation",  
stream);  
else  
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",  
"application/vnd.openxmlformats-officedocument.presentationml.presentation",  
stream);
```

By executing the program, you will get the PowerPoint slide as follows.



Add Date and Time in PowerPoint Slide

Essential Presentation library facilitates adding Date and Time to a slide of the PowerPoint Presentation. Date and Time comes with formatting options for date stamp as either it can be updated automatically using computer's clock or stay fixed until you change it.

The following code example demonstrates how to add Date and Time to a slide of the presentation.

C#

```
//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Sets the visibility of Date and Time in the slide
slide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the Date and Time to the Footer
slide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimehhmmssAMPM;
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```

'Creates an instance of Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds a blank slide
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Sets the visibility of Date and Time in the slide
slide.HeadersFooters.DateAndTime.Visible = True
'Sets the format of the Date and Time to the Footer
slide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimehmmssAMPM
'Adds textbox to the slide
Dim textboxShape As IShape = slide.AddTextBox(0, 0, 500, 500)
'Adds paragraph to the textbody of textbox
Dim paragraph As IParagraph = textboxShape.TextBody.AddParagraph()
'Adds a TextPart to the paragraph
Dim textPart As ITextPart = paragraph.AddTextPart()
'Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base."
'Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Sets the visibility of Date and Time in the slide
slide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the Date and Time to the Footer
slide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimehmmssAMPM;
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;

```

```

savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Sets the visibility of Date and Time in the slide
slide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the Date and Time to the Footer
slide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimehhmmssAMPM;
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbox of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Dispose the image stream
pictureStream.Dispose();
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Sets the visibility of Date and Time in the slide
slide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the Date and Time to the Footer
slide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimehhmmssAMPM;
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbox of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
//Adds a TextPart to the paragraph

```

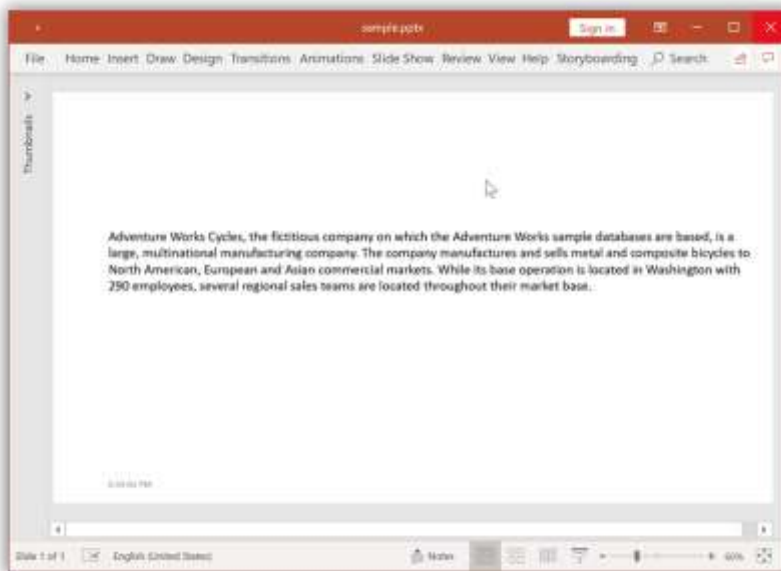


```

ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Create new memory stream to save Presentation
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Refer to the Presentation/Xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

By executing the program, you will get the PowerPoint slide as follows.



Add Slide Number to PowerPoint Slides

Essential Presentation library facilitates adding Slide number to a slide of the PowerPoint Presentation.

The following code example demonstrates how to add Slide number to a slide of the presentation.

C#

```

//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Sets the visibility of slide number in the slide
slide.HeadersFooters.SlideNumber.Visible = true;
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Creates an instance of Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds a blank slide
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Sets the visibility of slide number in the slide
slide.HeadersFooters.SlideNumber.Visible = True
'Adds textbox to the slide
Dim textboxShape As IShape = slide.AddTextBox(0, 0, 500, 500)
'Adds paragraph to the textbody of textbox
Dim paragraph As IParagraph = textboxShape.TextBody.AddParagraph()
'Adds a TextPart to the paragraph
Dim textPart As ITextPart = paragraph.AddTextPart()
'Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base."
'Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);

```

```

//Sets the visibility of slide number in the slide
slide.HeadersFooters.SlideNumber.Visible = true;
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

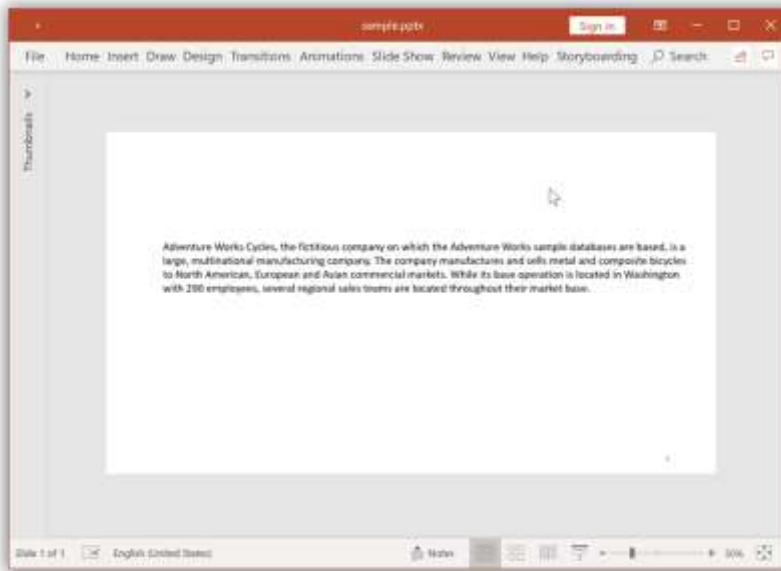
//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Sets the visibility of slide number in the slide
slide.HeadersFooters.SlideNumber.Visible = true;
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Dispose the image stream
pictureStream.Dispose();
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```
//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a blank slide
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Sets the visibility of slide number in the slide
slide.HeadersFooters.SlideNumber.Visible = true;
//Adds textbox to the slide
IShape textboxShape = slide.AddTextBox(0, 0, 500, 500);
//Adds paragraph to the textbody of textbox
IParagraph paragraph = textboxShape.TextBody.AddParagraph();
//Adds a TextPart to the paragraph
ITextPart textPart = paragraph.AddTextPart();
//Adds text to the TextPart
textPart.Text = "AdventureWorks Cycles, the fictitious company on which the
AdventureWorks sample databases are based, is a large, multinational
manufacturing company. The company manufactures and sells metal and
composite bicycles to North American, European and Asian commercial markets.
While its base operation is located in Washington with 290 employees,
several regional sales teams are located throughout their market base.";
//Create new memory stream to save Presentation
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Refer to the Presentation/Xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

By executing the program, you will get the PowerPoint slide as follows.



Add Footer to Master and Layout slides

Essential Presentation library facilitates adding Footers to both master and layout slides of the PowerPoint Presentation. If you want to use a different format for the Footers, then a great way to apply the format to all the slides (existing and new slides) is using the Slide Master and Slide layout. Under Slide Master and Slide layout, you can control the styles for footer options and this way you can apply different styles to the slides, or re-locate the Footer shape to any other position based on your requirement

The following code example demonstrates how to add a Footers to the master and layout slides of the presentation.

C#

```
//Load or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Footer.pptx");
//Access the first master slide in PowerPoint file
IMasterSlide masterSlide = pptxDoc.Masters[0];
//Sets the visibility of Footer content in the Master slide
masterSlide.HeadersFooters.Footer.Visible = true;
//Sets the text to be added to the Footer of the Master slide
masterSlide.HeadersFooters.Footer.Text = "Master Slide Footer";
//Sets the visibility of DateTime Footer in the Master slide
masterSlide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the DateTime Footer in the Master slide
masterSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimehhmmssAMPM;
//Iterate each layout slide in the Master slide
foreach(ILayoutSlide layoutSlide in masterSlide.LayoutSlides)
{
    //Sets the visibility of Footer content in the Layout slide
    layoutSlide.HeadersFooters.Footer.Visible = true;
    //Sets the text to be added to the Footer of the Layout slide
    layoutSlide.HeadersFooters.Footer.Text = "Layout slide Footer";
    //Sets the visibility of DateTime Footer in Layout slide
    layoutSlide.HeadersFooters.DateAndTime.Visible = true;
    //Sets the format of the DateTime Footer in Layout slide
```

```

layoutSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimeMMddyyhmmAMPM;
}
//Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Load or open an PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Footer.pptx")
'Access the first master slide in PowerPoint file
Dim masterSlide As IMasterSlide = pptxDoc.Masters(0)
'Sets the visibility of Footer content in the Master slide
masterSlide.HeadersFooters.Footer.Visible = True
'Sets the text to be added to the Footer of the Master slide
masterSlide.HeadersFooters.Footer.Text = "Master Slide Footer"
'Sets the visibility of DateTime Footer in the Master slide
masterSlide.HeadersFooters.DateAndTime.Visible = True
'Sets the format of the DateTime Footer in the Master slide
masterSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimehmmssAMPM
'Iterate each layout slide in the Master slide
For Each layoutSlide As ILayoutSlide In masterSlide.LayoutSlides
'Sets the visibility of Footer content in the Layout slide
layoutSlide.HeadersFooters.Footer.Visible = True
'Sets the text to be added to the Footer of the Layout slide
layoutSlide.HeadersFooters.Footer.Text = "Layout slide Footer"
'Sets the visibility of DateTime Footer in Layout slide
layoutSlide.HeadersFooters.DateAndTime.Visible = True
'Sets the format of the DateTime Footer in Layout slide
layoutSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimeMMddyyhmmAMPM
Next
'Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Access the first master slide in PowerPoint file
IMasterSlide masterSlide = pptxDoc.Masters[0];
//Sets the visibility of Footer content in the Master slide
masterSlide.HeadersFooters.Footer.Visible = true;
//Sets the text to be added to the Footer of the Master slide
masterSlide.HeadersFooters.Footer.Text = "Master Slide Footer";

```

```

//Sets the visibility of DateTime Footer in the Master slide
masterSlide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the DateTime Footer in the Master slide
masterSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimehhmmssAMPM;
//Iterate each layout slide in the Master slide
foreach(ILayoutSlide layoutSlide in masterSlide.LayoutSlides)
{
//Sets the visibility of Footer content in the Layout slide
layoutSlide.HeadersFooters.Footer.Visible = true;
//Sets the text to be added to the Footer of the Layout slide
layoutSlide.HeadersFooters.Footer.Text = "Layout slide Footer";
//Sets the visibility of DateTime Footer in Layout slide
layoutSlide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the DateTime Footer in Layout slide
layoutSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimeMMddyyhmmAMPM;
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Access the first master slide in PowerPoint file
IMasterSlide masterSlide = pptxDoc.Masters[0];
//Sets the visibility of Footer content in the Master slide
masterSlide.HeadersFooters.Footer.Visible = true;
//Sets the text to be added to the Footer of the Master slide
masterSlide.HeadersFooters.Footer.Text = "Master Slide Footer";
//Sets the visibility of DateTime Footer in the Master slide
masterSlide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the DateTime Footer in the Master slide
masterSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimehhmmssAMPM;
//Iterate each layout slide in the Master slide
foreach(ILayoutSlide layoutSlide in masterSlide.LayoutSlides)
{
//Sets the visibility of Footer content in the Layout slide
layoutSlide.HeadersFooters.Footer.Visible = true;
//Sets the text to be added to the Footer of the Layout slide
layoutSlide.HeadersFooters.Footer.Text = "Layout slide Footer";
//Sets the visibility of DateTime Footer in Layout slide
layoutSlide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the DateTime Footer in Layout slide

```

```

layoutSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimeMMddyyhmmAMPM;
}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Dispose the image stream
pictureStream.Dispose();
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

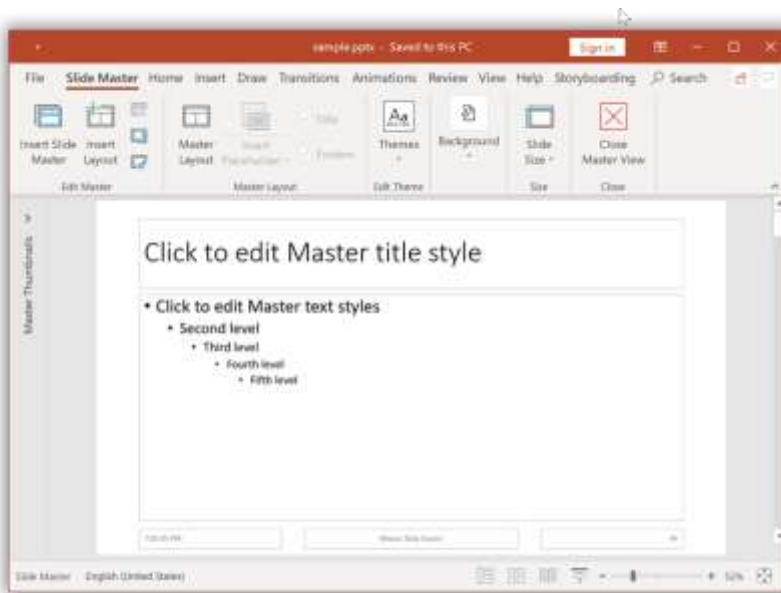
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Access the first master slide in PowerPoint file
IMasterSlide masterSlide = pptxDoc.Masters[0];
//Sets the visibility of Footer content in the Master slide
masterSlide.HeadersFooters.Footer.Visible = true;
//Sets the text to be added to the Footer of the Master slide
masterSlide.HeadersFooters.Footer.Text = "Master Slide Footer";
//Sets the visibility of DateTime Footer in the Master slide
masterSlide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the DateTime Footer in the Master slide
masterSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimehmmssAMPM;
//Iterate each layout slide in the Master slide
foreach (ILayoutSlide layoutSlide in masterSlide.LayoutSlides)
{
    //Sets the visibility of Footer content in the Layout slide
    layoutSlide.HeadersFooters.Footer.Visible = true;
    //Sets the text to be added to the Footer of the Layout slide
    layoutSlide.HeadersFooters.Footer.Text = "Layout slide Footer";
    //Sets the visibility of DateTime Footer in Layout slide
    layoutSlide.HeadersFooters.DateAndTime.Visible = true;
    //Sets the format of the DateTime Footer in Layout slide
    layoutSlide.HeadersFooters.DateAndTime.Format =
    DateTimeFormatType.DateTimeMMddyyhmmAMPM;
}
//Create new memory stream to save Presentation
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the Presentation/Xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)

```

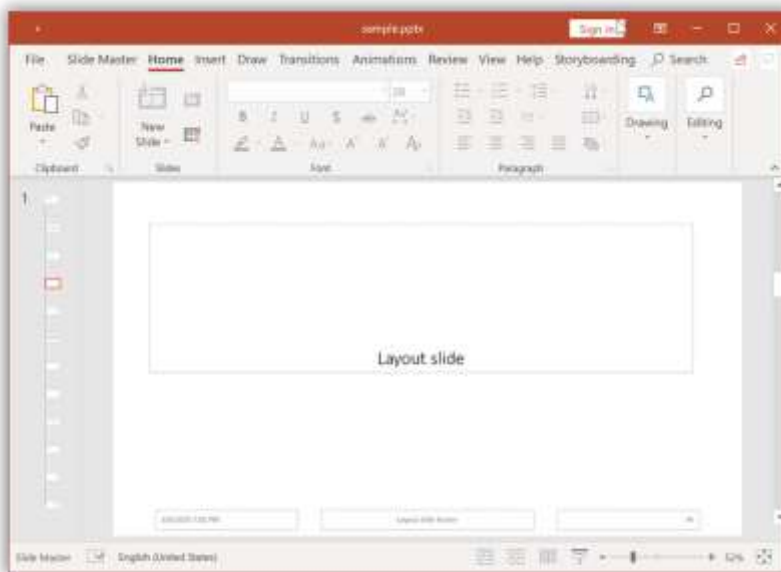


```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

By executing the program, you will get the PowerPoint Master slide as follows.



By executing the program, you will get the PowerPoint Layout slide as follows.



Add Headers and Footers into Notes slide

Essential Presentation library facilitates adding Headers and Footers to the Notes slide of the PowerPoint Presentation.

Note: 1. As per Microsoft PowerPoint behavior, you can add Header only in Notes slide of the PowerPoint using our Essential Presentation Library.

2. Header added in Notes slide will be visible only in the Notes page of the PowerPoint viewer.

The following code example demonstrates how to add a Headers and Footers to the Notes slide of the presentation.

C#

```
//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide
INotesSlide notesSlide = slide.AddNotesSlide();
//Sets the visibility of Header in the Notes slide
notesSlide.HeadersFooters.Header.Visible = true;
//Sets the text to be added to the Header of the Notes slide
notesSlide.HeadersFooters.Header.Text = "Header is added to Notes slide";
//Sets the visibility of DateTime Footer in the Notes slide
notesSlide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the DateTime Footer in the Notes slide
notesSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimeMMMy;
//Sets the visibility of Footer content in the Notes slide
notesSlide.HeadersFooters.Footer.Visible = true;
//Sets the text to be added to the Footer of the Notes slide
notesSlide.HeadersFooters.Footer.Text = "Notes slide Footer";
//Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Creates an instance of Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds new slide with blank slide layout type
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds new notes slide in the specified slide
Dim notesSlide As INotesSlide = slide.AddNotesSlide()
'Sets the visibility of Header in the Notes slide
notesSlide.HeadersFooters.Header.Visible = True
'Sets the text to be added to the Header of the Notes slide
notesSlide.HeadersFooters.Header.Text = "Header is added to Notes slide"
'Sets the visibility of DateTime Footer in the Notes slide
notesSlide.HeadersFooters.DateAndTime.Visible = True
'Sets the format of the DateTime Footer in the Notes slide
notesSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimeMMMy
'Sets the visibility of Footer content in the Notes slide
notesSlide.HeadersFooters.Footer.Visible = True
'Sets the text to be added to the Footer of the Notes slide
notesSlide.HeadersFooters.Footer.Text = "Notes slide Footer"
'Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx")
```

```
'Closes the Presentation
pptxDoc.Close()
```

UWP

```
//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide
INotesSlide notesSlide = slide.AddNotesSlide();
//Sets the visibility of Header in the Notes slide
notesSlide.HeadersFooters.Header.Visible = true;
//Sets the text to be added to the Header of the Notes slide
notesSlide.HeadersFooters.Header.Text = "Header is added to Notes slide";
//Sets the visibility of DateTime Footer in the Notes slide
notesSlide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the DateTime Footer in the Notes slide
notesSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimeMMMy;
//Sets the visibility of Footer content in the Notes slide
notesSlide.HeadersFooters.Footer.Visible = true;
//Sets the text to be added to the Footer of the Notes slide
notesSlide.HeadersFooters.Footer.Text = "Notes slide Footer";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

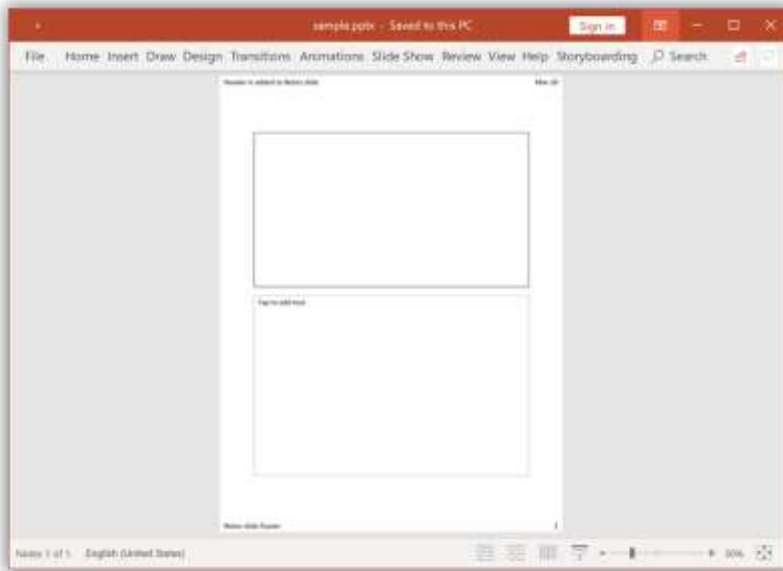
```
//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide
INotesSlide notesSlide = slide.AddNotesSlide();
//Sets the visibility of Header in the Notes slide
notesSlide.HeadersFooters.Header.Visible = true;
//Sets the text to be added to the Header of the Notes slide
notesSlide.HeadersFooters.Header.Text = "Header is added to Notes slide";
//Sets the visibility of DateTime Footer in the Notes slide
notesSlide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the DateTime Footer in the Notes slide
notesSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimeMMMy;
//Sets the visibility of Footer content in the Notes slide
notesSlide.HeadersFooters.Footer.Visible = true;
//Sets the text to be added to the Footer of the Notes slide
notesSlide.HeadersFooters.Footer.Text = "Notes slide Footer";
```

```
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Dispose the image stream
pictureStream.Dispose();
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```
//Creates an instance of Presentation
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide
INotesSlide notesSlide = slide.AddNotesSlide();
//Sets the visibility of Header in the Notes slide
notesSlide.HeadersFooters.Header.Visible = true;
//Sets the text to be added to the Header of the Notes slide
notesSlide.HeadersFooters.Header.Text = "Header is added to Notes slide";
//Sets the visibility of DateTime Footer in the Notes slide
notesSlide.HeadersFooters.DateAndTime.Visible = true;
//Sets the format of the DateTime Footer in the Notes slide
notesSlide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimeMMMy;
//Sets the visibility of Footer content in the Notes slide
notesSlide.HeadersFooters.Footer.Visible = true;
//Sets the text to be added to the Footer of the Notes slide
notesSlide.HeadersFooters.Footer.Text = "Notes slide Footer";
//Create new memory stream to save Presentation
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the Presentation/Xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

By executing the program, you will get the PowerPoint Notes slide as follows.



Modify Headers and Footers in PowerPoint

Edit Footer text of an existing Slide

Essential Presentation library facilitates editing the Footer text of an existing slide in the PowerPoint Presentation.

The following code example demonstrates how to edit Footer text of an existing slide in the presentation.

C#

```
//Load or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Footer.pptx");
//Gets the first slide from the cloned PowerPoint presentation
ISlide slide = pptxDoc.Slides[0];
//Modify the Footer text
slide.HeadersFooters.Footer.Text = "Footer content modified";
//Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Load or open an PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Footer.pptx")
'Gets the first slide from the PowerPoint presentation
ISlide slide = pptxDoc.Slides[0]
'Modify the Footer text
slide.HeadersFooters.Footer.Text = "Footer content modified"
'Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx")
'Closes the Presentation
pptxDoc.Close()
```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Load or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Modify the Footer text.
slide.HeadersFooters.Footer.Text = "Footer content modified";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Load or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide from the cloned PowerPoint presentation
ISlide slide = pptxDoc.Slides[0];
//Modify the Footer text
slide.HeadersFooters.Footer.Text = "Footer content modified";
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Dispose the image stream
pictureStream.Dispose();
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

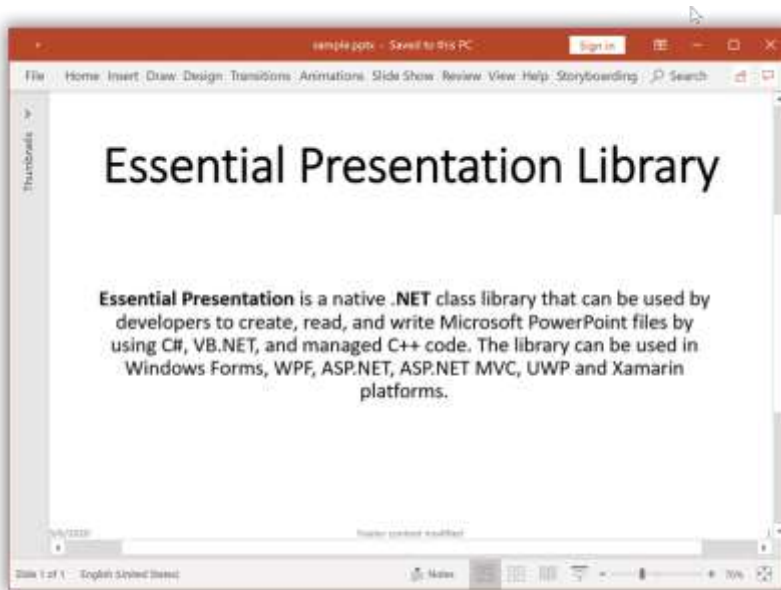
```

//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
//Load or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Modify the Footer text
slide.HeadersFooters.Footer.Text = "Footer content modified";
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;

```

```
//The operation in Save under Xamarin varies between Windows Phone, Android,
//and iOS platforms. Refer to the Presentation/Xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

By executing the program, you will get the PowerPoint slide as follows.



Modify Date and Time format of an existing Slide

Essential Presentation library facilitates modifying the Date and Time of an existing slide in the PowerPoint Presentation.

The following code example demonstrates how to modify Date and Time of an existing slide in the presentation.

C#

```
//Load or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Footer.pptx");
//Gets the first slide from the cloned PowerPoint presentation
ISlide slide = pptxDoc.Slides[0];
//Modify Date and Time format of the Footer
slide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTImeddddMMMMdyyyyy;
//Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Load or open an PowerPoint Presentation  
Dim pptxDoc As IPresentation = Presentation.Open("Footer.pptx")  
'Gets the first slide from the PowerPoint presentation  
ISlide slide = pptxDoc.Slides[0]  
'Modify Date and Time format of the Footer  
slide.HeadersFooters.DateAndTime.Format =  
DateTimeFormatType.DateTimeddddMMMMdyyyyy  
'Saves the Presentation to the file system  
pptxDoc.Save("Sample.pptx")  
'Closes the Presentation  
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker  
FileOpenPicker openPicker = new FileOpenPicker();  
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;  
openPicker.FileTypeFilter.Add(".pptx");  
//Creates a storage file from FileOpenPicker  
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();  
//Load or open an PowerPoint Presentation  
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);  
//Modify Date and Time format of the Footer  
slide.HeadersFooters.DateAndTime.Format =  
DateTimeFormatType.DateTimeddddMMMMdyyyyy;  
//Initializes FileSavePicker  
FileSavePicker savePicker = new FileSavePicker();  
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;  
savePicker.SuggestedFileName = "Output";  
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {  
    ".pptx" });  
//Creates a storage file from FileSavePicker  
StorageFile storageFile = await savePicker.PickSaveFileAsync();  
//Saves changes to the specified storage file  
await pptxDoc.SaveAsync(storageFile);
```

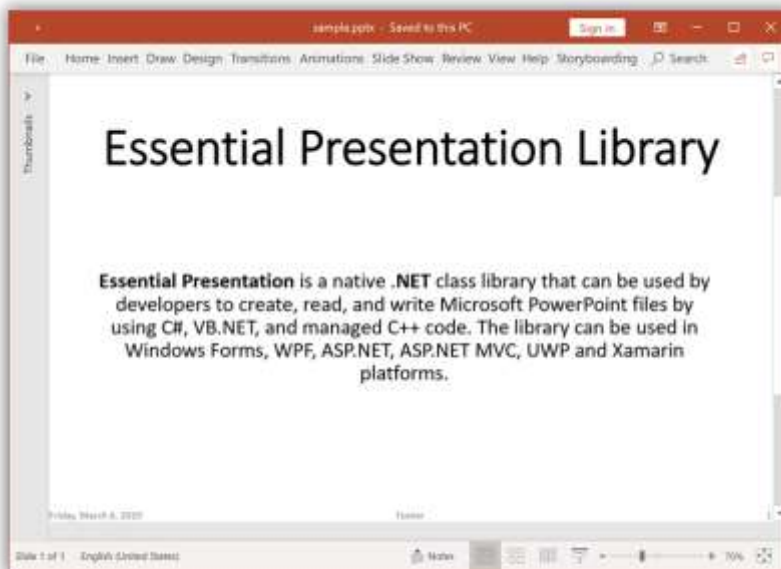
ASP.NET CORE

```
//Load or open an PowerPoint Presentation  
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);  
IPresentation pptxDoc = Presentation.Open(inputStream);  
//Gets the first slide from the cloned PowerPoint presentation  
ISlide slide = pptxDoc.Slides[0];  
//Modify Date and Time format of the Footer  
slide.HeadersFooters.DateAndTime.Format =  
DateTimeFormatType.DateTimeddddMMMMdyyyyy;  
//Save the PowerPoint Presentation as stream  
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);  
pptxDoc.Save(outputStream);  
//Dispose the image stream  
pictureStream.Dispose();  
//Closes the Presentation  
pptxDoc.Close();
```


XAMARIN

```
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
//Load or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Modify Date and Time format of the Footer
slide.HeadersFooters.DateAndTime.Format =
DateTimeFormatType.DateTimeddddMMMMddyyyy;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android,
and iOS platforms. Refer to the Presentation/Xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

By executing the program, you will get the PowerPoint slide as follows.

*Modify font of the Footer text*

Essential Presentation library facilitates editing font of the Footer content in slide of the PowerPoint Presentation.

The following code example demonstrates how to edit font of the Footer content in slide of the presentation.

C#

```
//Load or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Footer.pptx");
//Gets the first slide from the cloned PowerPoint presentation
ISlide slide = pptxDoc.Slides[0];
//Iterate each shape in slide
foreach (IShape shape in slide.Shapes)
{
    //Check whether the shape is with Placeholder SlideItemType and
    PlaceholderType as Footer
    if (shape.SlideItemType == SlideItemType.Placeholder &&
        shape.PlaceholderFormat.Type == PlaceholderType.Footer)
    {
        //Change the font name for the Footer content
        shape.TextBody.Paragraphs[0].Font.FontName = "Verdana";
        //Change the font size for the Footer content
        shape.TextBody.Paragraphs[0].Font.FontSize = 18;
    }
}
//Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Load or open an PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Footer.pptx")
'Gets the first slide from the PowerPoint presentation
ISlide slide = pptxDoc.Slides[0]
'Iterate each shape in slide
For Each shape As IShape In slide.Shapes
    'Check whether the shape is with Placeholder SlideItemType and
    PlaceholderType as Footer
    If shape.SlideItemType = SlideItemType.Placeholder AndAlso
        shape.PlaceholderFormat.Type = PlaceholderType.Footer Then
        'Change the font name for the Footer content
        shape.TextBody.Paragraphs(0).Font.FontName = "Verdana"
        'Change the font size for the Footer content
        shape.TextBody.Paragraphs(0).Font.FontSize = 18
    End If
Next
'Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx")
'Closes the Presentation
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
```

```

openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Gets the first slide from the cloned PowerPoint presentation
ISlide slide = pptxDoc.Slides[0];
//Iterate each shape in slide
foreach(IShape shape in slide.Shapes)
{
    //Check whether the shape is with Placeholder SlideItemType and
    PlaceholderType as Footer
    if (shape.SlideItemType == SlideItemType.Placeholder &&
        shape.PlaceholderFormat.Type == PlaceholderType.Footer)
    {
        //Change the font name for the Footer content
        shape.TextBody.Paragraphs[0].Font.FontName = "Verdana";
        //Change the font size for the Footer content
        shape.TextBody.Paragraphs[0].Font.FontSize = 18;
    }
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Load or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide from the cloned PowerPoint presentation
ISlide slide = pptxDoc.Slides[0];
//Iterate each shape in slide
foreach(IShape shape in slide.Shapes)
{
    //Check whether the shape is with Placeholder SlideItemType and
    PlaceholderType as Footer
    if (shape.SlideItemType == SlideItemType.Placeholder &&
        shape.PlaceholderFormat.Type == PlaceholderType.Footer)
    {
        //Change the font name for the Footer content
        shape.TextBody.Paragraphs[0].Font.FontName = "Verdana";
        //Change the font size for the Footer content
        shape.TextBody.Paragraphs[0].Font.FontSize = 18;
    }
}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);

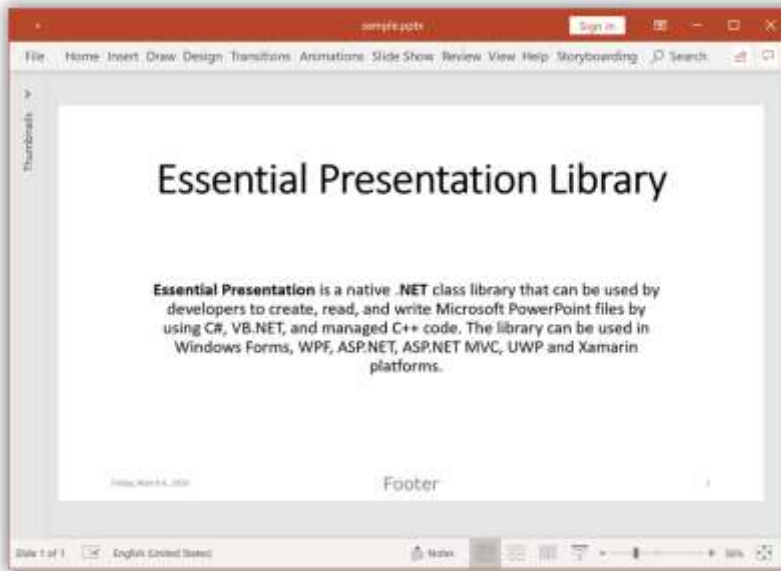
```

```
//Dispose the image stream
pictureStream.Dispose();
//Closes the Presentation
pptxDoc.Close();
```

XAMARIN

```
///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
///Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
///Gets the first slide from the cloned PowerPoint presentation
ISlide slide = pptxDoc.Slides[0];
///Iterate each shape in slide
foreach (IShape shape in slide.Shapes)
{
    ///Check whether the shape is with Placeholder SlideItemType and PlaceholderType as Footer
    if (shape.SlideItemType == SlideItemType.Placeholder &&
        shape.PlaceholderFormat.Type == PlaceholderType.Footer)
    {
        ///Change the font name for the Footer content
        shape.TextBody.Paragraphs[0].Font.FontName = "Verdana";
        ///Change the font size for the Footer content
        shape.TextBody.Paragraphs[0].Font.FontSize = 18;
    }
}
///Create new memory stream to save Presentation
MemoryStream stream = new MemoryStream();
///Save Presentation in stream format
pptxDoc.Save(stream);
///Close the presentation
pptxDoc.Close();
stream.Position = 0;
///The operation in Save under Xamarin varies between Windows Phone, Android, and iOS platforms. Refer to the Presentation/Xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
        "application/vnd.openxmlformats-officedocument.presentationml.presentation",
        stream);
```

By executing the program, you will get the PowerPoint slide as follows.



Remove Headers and Footers from Title Slides

Essential Presentation library facilitates removing Footers from all the Title slides in the PowerPoint Presentation.

The following code example demonstrates how to remove Footers from all the Title slides in the presentation.

C#

```
//Load or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Footer.pptx");
//Iterate each slide in the Presentation
foreach(ISlide slide in pptxDoc.Slides)
{
    //Checks whether the LayoutType of Layout slide is Title
    if (slide.LayoutSlide.LayoutType == SlideLayoutType.Title)
    {
        //Sets the visibility of DateAndTime in the Title slide
        slide.HeadersFooters.DateAndTime.Visible = false;
        //Sets the visibility of Footer in the Title slide
        slide.HeadersFooters.Footer.Visible = false;
        //Sets the visibility of SlideNumber in the Title slide
        slide.HeadersFooters.SlideNumber.Visible = false;
    }
}
//Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx");
//Closes the Presentation
pptxDoc.Close();
```

VB.NET

```
'Load or open an PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Footer.pptx")
'Iterate each slide in the Presentation
For Each slide As ISlide In pptxDoc.Slides
```

```

'Checks whether the LayoutType of Layout slide is Title
If slide.LayoutSlide.LayoutType = SlideLayoutType.Title Then
'Sets the visibility of DateAndTime in the Title slide
slide.HeadersFooters.DateAndTime.Visible = False
'Sets the visibility of Footer in the Title slide
slide.HeadersFooters.Footer.Visible = False
'Sets the visibility of SlideNumber in the Title slide
slide.HeadersFooters.SlideNumber.Visible = False
End If
Next
'Saves the Presentation to the file system
pptxDoc.Save("Sample.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Load or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Iterate each slide in the Presentation
foreach(ISlide slide in pptxDoc.Slides)
{
    //Checks whether the LayoutType of Layout slide is Title
    if (slide.LayoutSlide.LayoutType == SlideLayoutType.Title)
    {
        //Sets the visibility of DateAndTime in the Title slide
        slide.HeadersFooters.DateAndTime.Visible = false;
        //Sets the visibility of Footer in the Title slide
        slide.HeadersFooters.Footer.Visible = false;
        //Sets the visibility of SlideNumber in the Title slide
        slide.HeadersFooters.SlideNumber.Visible = false;
    }
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Load or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Iterate each slide in the Presentation

```

```

foreach(ISlide slide in pptxDoc.Slides)
{
    //Checks whether the LayoutType of Layout slide is Title
    if (slide.LayoutSlide.LayoutType == SlideLayoutType.Title)
    {
        //Sets the visibility of DateAndTime in the Title slide
        slide.HeadersFooters.DateAndTime.Visible = false;
        //Sets the visibility of Footer in the Title slide
        slide.HeadersFooters.Footer.Visible = false;
        //Sets the visibility of SlideNumber in the Title slide
        slide.HeadersFooters.SlideNumber.Visible = false;
    }
}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sample.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Dispose the image stream
pictureStream.Dispose();
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

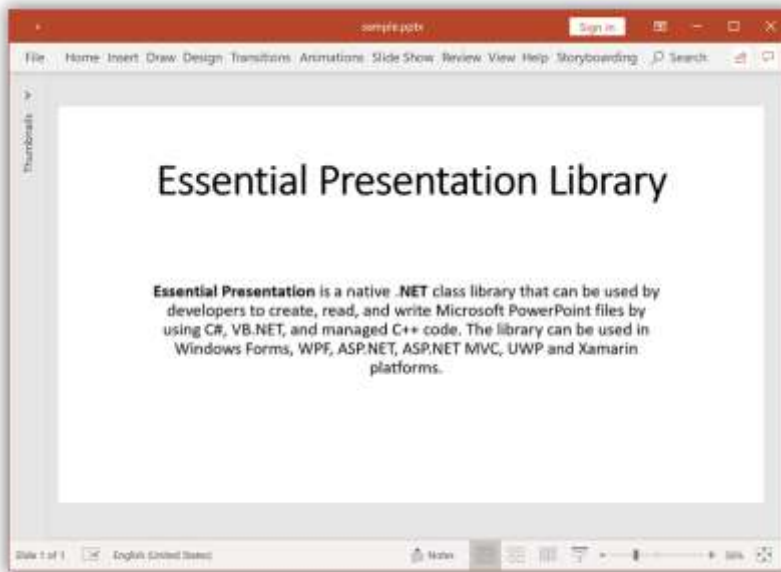
```

/"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
//Load or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Iterate each slide in the Presentation
foreach(ISlide slide in pptxDoc.Slides)
{
    //Checks whether the LayoutType of Layout slide is Title
    if (slide.LayoutSlide.LayoutType == SlideLayoutType.Title)
    {
        //Sets the visibility of DateAndTime in the Title slide
        slide.HeadersFooters.DateAndTime.Visible = false;
        //Sets the visibility of Footer in the Title slide
        slide.HeadersFooters.Footer.Visible = false;
        //Sets the visibility of SlideNumber in the Title slide
        slide.HeadersFooters.SlideNumber.Visible = false;
    }
}
//Create new memory stream to save Presentation
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android, and iOS platforms. Refer to the Presentation/Xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

By executing the program, you will get the PowerPoint slide as follows.



Working with Notes

Notes are the contents associated with each slide and are visible only to the presenter when monitors are shared in “Presenter View”. It shows hint for the speaker, so it is often called as “Speaker Notes”. The presenter can optionally add key points to notes. You can add and modify the notes in your slide using Essential Presentation library.

Adding Notes to a Slide

The below code example demonstrates how to create a Notes in a PowerPoint Slide.

C#

```
//Creates a Presentation without slides.
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide.
INotesSlide notesSlide = slide.AddNotesSlide();
//Adds text content into the Notes Slide.
notesSlide.NotesTextBody.AddParagraph("Notes content");
//Saves Presentation with specified file name with extension.
pptxDoc.Save("PresentationWithNotesSlide.pptx");
```

VB.NET

```
'Creates a Presentation without slides.
```



```

Dim pptxDoc As IPresentation = Presentation.Create()
'Adds new slide with blank slide layout type.
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds new notes slide in the specified slide.
Dim notesSlide As INotesSlide = slide.AddNotesSlide()
'Adds text content into the Notes Slide.
notesSlide.NotesTextBody.AddParagraph("Notes content")
'Saves Presentation with specified file name with extension.
pptxDoc.Save("PresentationWithNotesSlide.pptx")

```

UWP

```

//Creates a Presentation without slides.
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide.
INotesSlide notesSlide = slide.AddNotesSlide();
//Adds text content into the Notes Slide.
notesSlide.NotesTextBody.AddParagraph("Notes content");
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "PresentationWithNotesSlide";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates a Presentation without slides.
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide.
INotesSlide notesSlide = slide.AddNotesSlide();
//Adds text content into the Notes Slide.
notesSlide.NotesTextBody.AddParagraph("Notes content");
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputStreamName, FileMode.Create);
pptxDoc.Save(outputStream);

```

XAMARIN

```

//Creates a Presentation without slides.
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide.
INotesSlide notesSlide = slide.AddNotesSlide();
//Adds text content into the Notes Slide.
notesSlide.NotesTextBody.AddParagraph("Notes content");

```

```

//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Adding Text into the Notes

The following code example demonstrates how to add a text in a Notes.

C#

```

//Creates a Presentation without slides.
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide.
INotesSlide notesSlide = slide.AddNotesSlide();
//Adds Paragraph into the text body.
IParagraph paragraph = notesSlide.NotesTextBody.AddParagraph();
//Adds text part into the Paragraph.
ITextPart textPart = paragraph.AddTextPart();
textPart.Text = "The notes slide represents the contents and key notes of
the corresponding slide. It is more useful when we use Presenter View while
presenting the seminars through SlideShow.";
//Sets Bold format for text content.
textPart.Font.Bold=true;
// Sets font style using font name.
textPart.Font.FontName = "Times New Roman";
// Sets text content size using FontSize property.
textPart.Font.FontSize = 20;
//Saves Presentation with specified file name with extension.
pptxDoc.Save("PresentationWithNotesSlide.pptx");

```

VB.NET

```

'Creates a Presentation without slides.
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds new slide with blank slide layout type.
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds new notes slide in the specified slide.
Dim notesSlide As INotesSlide = slide.AddNotesSlide()
'Adds Paragraph into the text body.

```

```

Dim paragraph As IParagraph = notesSlide.NotesTextBody.AddParagraph()
'Adds text part into the Paragraph.
Dim textPart As ITextPart = paragraph.AddTextPart()
textPart.Text = "The notes slide represents the contents and key notes of
the corresponding slide. It is more useful when we use Presenter View while
presenting the seminars through SlideShow."
'Sets Bold format for text content.
textPart.Font.Bold = True
'Sets font style using font name.
textPart.Font.FontName = "Times New Roman"
'Sets text content size using FontSize property.
textPart.Font.FontSize = 20
'Saves Presentation with specified file name with extension.
pptxDoc.Save("PresentationWithNotesSlide.pptx")

```

UWP

```

//Creates a Presentation without slides.
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide.
INotesSlide notesSlide = slide.AddNotesSlide();
//Adds Paragraph into the text body.
IParagraph paragraph = notesSlide.NotesTextBody.AddParagraph();
//Adds text part into the Paragraph.
ITextPart textPart = paragraph.AddTextPart();
textPart.Text = "The notes slide represents the contents and key notes of
the corresponding slide. It is more useful when we use Presenter View while
presenting the seminars through SlideShow.";
//Sets Bold format for text content.
textPart.Font.Bold=true;
//Sets font style using font name.
textPart.Font.FontName = "Times New Roman";
//Sets text content size using FontSize property.
textPart.Font.FontSize = 20;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "PresentationWithNotesSlide";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates a Presentation without slides.
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide.
INotesSlide notesSlide = slide.AddNotesSlide();
//Adds Paragraph into the text body.

```

```

IParagraph paragraph = notesSlide.NotesTextBody.AddParagraph();
//Adds text part into the Paragraph.
ITextPart textPart = paragraph.AddTextPart();
textPart.Text = "The notes slide represents the contents and key notes of
the corresponding slide. It is more useful when we use Presenter View while
presenting the seminars through SlideShow.";
//Sets Bold format for text content.
textPart.Font.Bold=true;
// Sets font style using font name.
textPart.Font.FontName = "Times New Roman";
// Sets text content size using FontSize property.
textPart.Font.FontSize = 20;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);

```

XAMARIN

```

//Creates a Presentation without slides.
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide.
INotesSlide notesSlide = slide.AddNotesSlide();
//Adds Paragraph into the text body.
IParagraph paragraph = notesSlide.NotesTextBody.AddParagraph();
//Adds text part into the Paragraph.
ITextPart textPart = paragraph.AddTextPart();
textPart.Text = "The notes slide represents the contents and key notes of
the corresponding slide. It is more useful when we use Presenter View while
presenting the seminars through SlideShow.";
//Sets Bold format for text content.
textPart.Font.Bold=true;
// Sets font style using font name.
textPart.Font.FontName = "Times New Roman";
// Sets text content size using FontSize property.
textPart.Font.FontSize = 20;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Adding a numbered list to Notes

The following code example demonstrates how to create simple numbered list as Notes.

C#

```
//Creates a Presentation without slides.
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide.
INotesSlide notesSlide = slide.AddNotesSlide();
// Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = notesSlide.NotesTextBody.AddParagraph("The Northwind
sample database (Northwind.mdb) is included with all versions of Access.");
//Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the starting value as 1
paragraph.ListFormat.StartValue = 1;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = notesSlide.NotesTextBody.AddParagraph("It provides data you can
experiment with and database objects that demonstrate features you might
want to implement in your own databases.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = notesSlide.NotesTextBody.AddParagraph("Using Northwind, you can
become familiar with how a relational database is structured and how the
database objects work together to help you enter, store, manipulate, and
print your data.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
```

```

paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Saves the Presentation to the file system.
pptxDoc.Save("Sample.pptx");
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Creates a Presentation without slides.
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds new slide with blank slide layout type.
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds new notes slide in the specified slide.
Dim notesSlide As INotesSlide = slide.AddNotesSlide()
' Adds a new paragraph with the text in the left hand side textbox.
Dim paragraph As IParagraph = notesSlide.NotesTextBody.AddParagraph("The
Northwind sample database (Northwind.mdb) is included with all versions of
Access.")
'Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Numbered
'Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod
'Sets the starting value as 1
paragraph.ListFormat.StartValue = 1
'Sets the list level as 1
paragraph.IndentLevelNumber = 1
' Sets the hanging value
paragraph.FirstLineIndent = -20
' Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
paragraph.ListFormat.Size = 100
' Adds another paragraph with the text in the left hand side textbox.
paragraph = notesSlide.NotesTextBody.AddParagraph("It provides data you can
experiment with and database objects that demonstrate features you might
want to implement in your own databases.")
'Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered
'Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod
'Sets the list level as 1
paragraph.IndentLevelNumber = 1
' Sets the hanging value
paragraph.FirstLineIndent = -20
' Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
paragraph.ListFormat.Size = 100
' Adds another paragraph with the text in the left hand side textbox.
paragraph = notesSlide.NotesTextBody.AddParagraph("Using Northwind, you can
become familiar with how a relational database is structured and how the

```

```

database objects work together to help you enter, store, manipulate, and
print your data.")
'Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered
'Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod
'Sets the list level as 1
paragraph.IndentLevelNumber = 1
' Sets the hanging value
paragraph.FirstLineIndent = -20
' Sets the bullet character size. Here, 100 means 100% of its text. Possible
values can range from 25 to 400.
paragraph.ListFormat.Size = 100
'Saves the Presentation to the file system.
pptxDoc.Save("Sample.pptx")
'Closes the Presentation
pptxDoc.Close()

```

UWP

```

//Creates a Presentation without slides.
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide.
INotesSlide notesSlide = slide.AddNotesSlide();
// Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = notesSlide.NotesTextBody.AddParagraph("The Northwind
sample database (Northwind.mdb) is included with all versions of Access.");
//Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the starting value as 1
paragraph.ListFormat.StartValue = 1;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = notesSlide.NotesTextBody.AddParagraph("It provides data you can
experiment with and database objects that demonstrate features you might
want to implement in your own databases.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value

```

```

paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = notesSlide.NotesTextBody.AddParagraph("Using Northwind, you can
become familiar with how a relational database is structured and how the
database objects work together to help you enter, store, manipulate, and
print your data.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Creates a Presentation without slides.
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide.
INotesSlide notesSlide = slide.AddNotesSlide();
// Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = notesSlide.NotesTextBody.AddParagraph("The Northwind
sample database (Northwind.mdb) is included with all versions of Access.");
//Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the starting value as 1
paragraph.ListFormat.StartValue = 1;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.

```



```

paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = notesSlide.NotesTextBody.AddParagraph("It provides data you can
experiment with and database objects that demonstrate features you might
want to implement in your own databases.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = notesSlide.NotesTextBody.AddParagraph("Using Northwind, you can
become familiar with how a relational database is structured and how the
database objects work together to help you enter, store, manipulate, and
print your data.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the Presentation
pptxDoc.Close();

```

XAMARIN

```

//Creates a Presentation without slides.
IPresentation pptxDoc = Presentation.Create();
//Adds new slide with blank slide layout type.
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds new notes slide in the specified slide.
INotesSlide notesSlide = slide.AddNotesSlide();
// Adds a new paragraph with the text in the left hand side textbox.
IParagraph paragraph = notesSlide.NotesTextBody.AddParagraph("The Northwind
sample database (Northwind.mdb) is included with all versions of Access.");
//Sets the list type as Numbered
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;

```

```
//Sets the starting value as 1
paragraph.ListFormat.StartValue = 1;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
//Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = notesSlide.NotesTextBody.AddParagraph("It provides data you can
experiment with and database objects that demonstrate features you might
want to implement in your own databases.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
//Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
// Adds another paragraph with the text in the left hand side textbox.
paragraph = notesSlide.NotesTextBody.AddParagraph("Using Northwind, you can
become familiar with how a relational database is structured and how the
database objects work together to help you enter, store, manipulate, and
print your data.");
//Sets the list type as bulleted
paragraph.ListFormat.Type = ListType.Numbered;
//Sets the numbered style (list numbering) as Arabic number following by
period.
paragraph.ListFormat.NumberStyle = NumberedListStyle.ArabicPeriod;
//Sets the list level as 1
paragraph.IndentLevelNumber = 1;
// Sets the hanging value
paragraph.FirstLineIndent = -20;
// Sets the bullet character size. Here, 100 means 100% of its text.
//Possible values can range from 25 to 400.
paragraph.ListFormat.Size = 100;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

```
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Removing Notes from a Slide

The below code example demonstrates how to remove a Notes from a PowerPoint Slide.

C#

```
//Opens an existing PowerPoint presentation.
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Gets instance of the first slide from the Presentation.
ISlide slide = pptxDoc.Slides[0] as ISlide;
//Removes Notes Slide from a corresponding slide.
slide.RemoveNotesSlide();
//Saves Presentation with specified file name with extension.
pptxDoc.Save("PresentationWithNotesSlide.pptx");
```

VB.NET

```
'Opens an existing PowerPoint presentation.
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Gets instance of the first slide from the Presentation.
Dim slide As ISlide = TryCast(pptxDoc.Slides(0), ISlide)
'Removes Notes Slide from a corresponding slide.
slide.RemoveNotesSlide()
'Saves Presentation with specified file name with extension.
pptxDoc.Save("PresentationWithNotesSlide.pptx")
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Gets instance of the first slide from the Presentation.
ISlide slide = pptxDoc.Slides[0] as ISlide;
//Removes Notes Slide from a corresponding slide.
slide.RemoveNotesSlide();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "PresentationWithNotesSlide";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets instance of the first slide from the Presentation.
ISlide slide = pptxDoc.Slides[0] as ISlide;
//Removes Notes Slide from a corresponding slide.
slide.RemoveNotesSlide();
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream(OutputFileName, FileMode.Create);
pptxDoc.Save(outputStream);
```

XAMARIN

```
///"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(resourcePath);
///Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
///Gets instance of the first slide from the Presentation.
ISlide slide = pptxDoc.Slides[0] as ISlide;
///Removes Notes Slide from a corresponding slide.
slide.RemoveNotesSlide();
///Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
///Save Presentation in stream format.
pptxDoc.Save(stream);
///Close the presentation
pptxDoc.Close();
stream.Position = 0;
///The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sample.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Sample.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
```

Working with SmartArt

A SmartArt diagram is a visual representation of your information, to effectively communicate your ideas in presentations. You can add and modify the SmartArt diagrams in PowerPoint presentations using Essential Presentation library.

Adding SmartArt to a Slide

You can add any of the predefined SmartArt diagrams to PowerPoint Presentation. The following code example demonstrates adding a SmartArt to a Slide.

C#

```
// Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a BasicBlockList SmartArt to the slide at the specified size and
position.
ISmartArt smartArt = slide.Shapes.AddSmartArt(SmartArtType.BasicBlockList,
0, 0, 640, 426);
//Save the Presentation
pptxDoc.Save("SmartArt.pptx");
//Close the Presentation
pptxDoc.Close();
```

VB.NET

```
'Create an instance of PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Add a blank slide to the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a BasicBlockList SmartArt to the slide at the specified size and
position.
Dim smartArt As ISmartArt =
slide.Shapes.AddSmartArt(SmartArtType.BasicBlockList, 0, 0, 640, 426)
'Save the Presentation
pptxDoc.Save("SmartArt.pptx")
'Close the Presentation
pptxDoc.Close()
```

UWP

```
//Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a BasicBlockList SmartArt to the slide at the specified size and
position.
ISmartArt smartArt = slide.Shapes.AddSmartArt(SmartArtType.BasicBlockList,
0, 0, 640, 426);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "SmartArt";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
// Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
```

```
//Add a BasicBlockList SmartArt to the slide at the specified size and
position.
ISmartArt smartArt = slide.Shapes.AddSmartArt(SmartArtType.BasicBlockList,
0, 0, 640, 426);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("SmartArt.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the Presentation
pptxDoc.Close();
```

XAMARIN

```
// Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a BasicBlockList SmartArt to the slide at the specified size and
position.
ISmartArt smartArt = slide.Shapes.AddSmartArt(SmartArtType.BasicBlockList,
0, 0, 640, 426);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("SmartArt.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("SmartArt.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Adding a node to the SmartArt

You can add a new node to the SmartArt diagram. The following code example demonstrates the same.

C#

```
// Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a SmartArt to the slide at the specified size and position
ISmartArt smartArt =
slide.Shapes.AddSmartArt(SmartArtType.AlternatingHexagons, 0, 0, 640, 426);
// Add a new node to the SmartArt.
ISmartArtNode newNode = smartArt.Nodes.Add();
// Set the text to the newly added node.
newNode.TextBody.AddParagraph("New main node added.");
```

```
//Save the Presentation.
pptxDoc.Save("SmartArt.pptx");
//Close the Presentation.
pptxDoc.Close();
```

VB.NET

```
'Create an instance of PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Add a blank slide to the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a SmartArt to the slide at the specified size and position
Dim smartArt As ISmartArt =
slide.Shapes.AddSmartArt(SmartArtType.AlternatingHexagons, 0, 0, 640, 426)
'Add a new node to the SmartArt.
Dim newNode As ISmartArtNode = smartArt.Nodes.Add()
'Set the text to the newly added node.
newNode.TextBody.AddParagraph("New main node added.")
'Save the Presentation.
pptxDoc.Save("SmartArt.pptx")
'Close the Presentation.
pptxDoc.Close()
```

UWP

```
//Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a SmartArt to the slide at the specified size and position
ISmartArt smartArt =
slide.Shapes.AddSmartArt(SmartArtType.AlternatingHexagons, 0, 0, 640, 426);
// Add a new node to the SmartArt.
ISmartArtNode newNode = smartArt.Nodes.Add();
// Set the text to the newly added node.
newNode.TextBody.AddParagraph("New main node added.");
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "SmartArt";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
// Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a SmartArt to the slide at the specified size and position
```

```

ISmartArt smartArt =
slide.Shapes.AddSmartArt(SmartArtType.AlternatingHexagons, 0, 0, 640, 426);
// Add a new node to the SmartArt.
ISmartArtNode newNode = smartArt.Nodes.Add();
// Set the text to the newly added node.
newNode.TextBody.AddParagraph("New main node added.");
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("SmartArt.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the Presentation.
pptxDoc.Close();

```

XAMARIN

```

// Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a SmartArt to the slide at the specified size and position
ISmartArt smartArt =
slide.Shapes.AddSmartArt(SmartArtType.AlternatingHexagons, 0, 0, 640, 426);
// Add a new node to the SmartArt.
ISmartArtNode newNode = smartArt.Nodes.Add();
// Set the text to the newly added node.
newNode.TextBody.AddParagraph("New main node added.");
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("SmartArt.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("SmartArt.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

In SmartArt diagrams, you can also add nodes to several nested levels. The maximum limit of nested levels may vary based on SmartArt types. The following code example demonstrates adding nested level nodes in a SmartArt.

C#

```

// Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);

```



```
//Add a SmartArt to the slide at the specified size and position.
ISmartArt smartArt =
slide.Shapes.AddSmartArt(SmartArtType.AlternatingHexagons, 0, 0, 640, 426);
// Add a new node to the SmartArt.
ISmartArtNode newNode = smartArt.Nodes.Add();
// Add a child node to the SmartArt node
ISmartArtNode childNode = newNode.ChildNodes.Add();
// Set a text to newly added child node.
childNode.TextBody.AddParagraph("Child node of the existing node.");
//Save the Presentation.
pptxDoc.Save("SmartArt.pptx");
//Close the Presentation.
pptxDoc.Close();
```

VB.NET

```
'Create an instance of PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Add a blank slide to the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a SmartArt to the slide at the specified size and position.
Dim smartArt As ISmartArt =
slide.Shapes.AddSmartArt(SmartArtType.AlternatingHexagons, 0, 0, 640, 426)
'Add a new node to the SmartArt.
Dim newNode As ISmartArtNode = smartArt.Nodes.Add()
'Add a child node to the SmartArt node
Dim childNode As ISmartArtNode = newNode.ChildNodes.Add()
'Set a text to newly added child node.
childNode.TextBody.AddParagraph("Child node of the existing node.")
'Save the Presentation.
pptxDoc.Save("SmartArt.pptx")
'Close the Presentation.
pptxDoc.Close()
```

UWP

```
//Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a SmartArt to the slide at the specified size and position.
ISmartArt smartArt =
slide.Shapes.AddSmartArt(SmartArtType.AlternatingHexagons, 0, 0, 640, 426);
// Add a new node to the SmartArt.
ISmartArtNode newNode = smartArt.Nodes.Add();
// Add a child node to the SmartArt node
ISmartArtNode childNode = newNode.ChildNodes.Add();
// Set a text to newly added child node.
childNode.TextBody.AddParagraph("Child node of the existing node.");
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "SmartArt";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
```

```
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
// Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a SmartArt to the slide at the specified size and position.
ISmartArt smartArt =
slide.Shapes.AddSmartArt(SmartArtType.AlternatingHexagons, 0, 0, 640, 426);
// Add a new node to the SmartArt.
ISmartArtNode newNode = smartArt.Nodes.Add();
// Add a child node to the SmartArt node
ISmartArtNode childNode = newNode.ChildNodes.Add();
// Set a text to newly added child node.
childNode.TextBody.AddParagraph("Child node of the existing node.");
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("SmartArt.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the Presentation.
pptxDoc.Close();
```

XAMARIN

```
// Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a SmartArt to the slide at the specified size and position.
ISmartArt smartArt =
slide.Shapes.AddSmartArt(SmartArtType.AlternatingHexagons, 0, 0, 640, 426);
// Add a new node to the SmartArt.
ISmartArtNode newNode = smartArt.Nodes.Add();
// Add a child node to the SmartArt node
ISmartArtNode childNode = newNode.ChildNodes.Add();
// Set a text to newly added child node.
childNode.TextBody.AddParagraph("Child node of the existing node.");
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("SmartArt.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("SmartArt.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Modifying SmartArt appearance

You can modify the SmartArt appearance by modifying the fill type, color, transparency etc. The below code example demonstrates modifying the appearance of SmartArt nodes.

C#

```
//Open a PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("SampleDocument.pptx");
//Get the Slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the SmartArt from Slide.
ISmartArt smartArt = slide.Shapes[0] as ISmartArt;
//Get the first node
ISmartArtNode firstNode = smartArt.Nodes[0];
// Set the text content of node.
firstNode.TextBody.AddParagraph("First Node");
//Set the fill type of node.
firstNode.Shapes[0].Fill.FillType = FillType.Solid;
// Set the fill color of node.
firstNode.Shapes[0].Fill.SolidFill.Color = ColorObject.GreenYellow;
//Set transparency value of fill
firstNode.Shapes[0].Fill.SolidFill.Transparency = 30;
//Save the Presentation.
pptxDoc.Save("SmartArt.pptx");
//Close the Presentation.
pptxDoc.Close();
```

VB.NET

```
'Open a PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("SampleDocument.pptx")
'Get the Slide from Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Get the SmartArt from Slide.
Dim smartArt As ISmartArt = TryCast(slide.Shapes(0), ISmartArt)
'Get the first node
Dim firstNode As ISmartArtNode = smartArt.Nodes(0)
' Set the text content of node.
firstNode.TextBody.AddParagraph("First Node")
'Set the fill type of node.
firstNode.Shapes(0).Fill.FillType = FillType.Solid
' Set the fill color of node.
firstNode.Shapes(0).Fill.SolidFill.Color = ColorObject.GreenYellow
'Set transparency value of fill
firstNode.Shapes(0).Fill.SolidFill.Transparency = 30
'Save the Presentation.
pptxDoc.Save("SmartArt.pptx")
'Close the Presentation.
pptxDoc.Close()
```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Get the Slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the SmartArt from Slide.
ISmartArt smartArt = slide.Shapes[0] as ISmartArt;
//Get the first node
ISmartArtNode firstNode = smartArt.Nodes[0];
// Set the text content of node.
firstNode.TextBody.AddParagraph("First Node");
//Set the fill type of node.
firstNode.Shapes[0].Fill.FillType = FillType.Solid;
// Set the fill color of node.
firstNode.Shapes[0].Fill.SolidFill.Color = ColorObject.GreenYellow;
//Set transparency value of fill
firstNode.Shapes[0].Fill.SolidFill.Transparency = 30;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "SmartArt";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get the Slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the SmartArt from Slide.
ISmartArt smartArt = slide.Shapes[0] as ISmartArt;
//Get the first node
ISmartArtNode firstNode = smartArt.Nodes[0];
// Set the text content of node.
firstNode.TextBody.AddParagraph("First Node");
//Set the fill type of node.
firstNode.Shapes[0].Fill.FillType = FillType.Solid;
// Set the fill color of node.
firstNode.Shapes[0].Fill.SolidFill.Color = ColorObject.GreenYellow;
//Set transparency value of fill
firstNode.Shapes[0].Fill.SolidFill.Transparency = 30;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("SmartArt.pptx", FileMode.Create);

```

```
pptxDoc.Save(outputStream);
//Close the Presentation.
pptxDoc.Close();
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get the Slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the SmartArt from Slide.
ISmartArt smartArt = slide.Shapes[0] as ISmartArt;
//Get the first node
ISmartArtNode firstNode = smartArt.Nodes[0];
// Set the text content of node.
firstNode.TextBody.AddParagraph("First Node");
//Set the fill type of node.
firstNode.Shapes[0].Fill.FillType = FillType.Solid;
// Set the fill color of node.
firstNode.Shapes[0].Fill.SolidFill.Color = ColorObject.GreenYellow;
//Set transparency value of fill
firstNode.Shapes[0].Fill.SolidFill.Transparency = 30;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("SmartArt.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("SmartArt.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Iterating through child nodes of an existing SmartArt

You can iterate through the child nodes and access the properties of each node in a SmartArt. The following code example demonstrates accessing and modifying the text content of node.

C#

```
//Open a PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("SampleDocument.pptx");
```

```

//Traverse through shape in the first slide.
foreach (IShape shape in pptxDoc.Slides[0].Shapes)
{
    if (shape is ISmartArt)
    {
        //Traverse through all nodes inside SmartArt
        foreach (ISmartArtNode mainNode in (shape as ISmartArt).Nodes)
        {
            if (mainNode.TextBody.Text == "Old Content")
            //Change the node content
            mainNode.TextBody.Paragraphs[0].TextParts[0].Text = "New Content";
        }
    }
}
//Save the Presentation.
pptxDoc.Save("SmartArt.pptx");
//Close the Presentation.
pptxDoc.Close();

```

VB.NET

```

'Open a PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("SampleDocument.pptx")
'Traverse through shape in the first slide.
For Each shape As IShape In pptxDoc.Slides(0).Shapes
    If TypeOf shape Is ISmartArt Then
        'Traverse through all nodes inside SmartArt
        For Each mainNode As ISmartArtNode In TryCast(shape, ISmartArt).Nodes
            If mainNode.TextBody.Text = "Old Content" Then
                'Change the node content
                mainNode.TextBody.Paragraphs(0).TextParts(0).Text = "New Content"
            End If
        Next
    End If
Next
'Save the Presentation.
pptxDoc.Save("SmartArt.pptx")
'Close the Presentation.
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Traverse through shape in the first slide.
foreach (IShape shape in pptxDoc.Slides[0].Shapes)
{
    if (shape is ISmartArt)
    {
        //Traverse through all nodes inside SmartArt

```

```

foreach (ISmartArtNode mainNode in (shape as ISmartArt).Nodes)
{
    if (mainNode.TextBody.Text == "Old Content")
    //Change the node content
    mainNode.TextBody.Paragraphs[0].TextParts[0].Text = "New Content";
}
}
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "SmartArt";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Traverse through shape in the first slide.
foreach (IShape shape in pptxDoc.Slides[0].Shapes)
{
    if (shape is ISmartArt)
    {
        //Traverse through all nodes inside SmartArt
        foreach (ISmartArtNode mainNode in (shape as ISmartArt).Nodes)
        {
            if (mainNode.TextBody.Text == "Old Content")
            //Change the node content
            mainNode.TextBody.Paragraphs[0].TextParts[0].Text = "New Content";
        }
    }
}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("SmartArt.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the Presentation.
pptxDoc.Close();

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Traverse through shape in the first slide.
foreach (IShape shape in pptxDoc.Slides[0].Shapes)
{

```

```

if (shape is ISmartArt)
{
    //Traverse through all nodes inside SmartArt
    foreach (ISmartArtNode mainNode in (shape as ISmartArt).Nodes)
    {
        if (mainNode.TextBody.Text == "Old Content")
        //Change the node content
        mainNode.TextBody.Paragraphs[0].TextParts[0].Text = "New Content";
    }
}

//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("SmartArt.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("SmartArt.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Removing node from an existing SmartArt

You can remove a node from the SmartArt diagram. The following code example demonstrates the same.

C#

```

//Open a PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("SampleDocument.pptx");
//Get the first slide from the Presentation.
ISlide slide = pptxDoc.Slides[0];
//Get the SmartArt from slide.
ISmartArt smartArt = slide.Shapes[0] as ISmartArt;
//Remove a node at the specified index.
smartArt.Nodes.RemoveAt(4);
//Save the Presentation.
pptxDoc.Save("SmartArt.pptx");
//Close the Presentation.
pptxDoc.Close();

```

VB.NET

```

'Open a PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("SampleDocument.pptx")
'Get the first slide from the Presentation.

```



```

Dim slide As ISlide = pptxDoc.Slides(0)
'Get the SmartArt from slide.
Dim smartArt As ISmartArt = TryCast(slide.Shapes(0), ISmartArt)
'Remove a node at the specified index.
smartArt.Nodes.RemoveAt(4)
'Save the Presentation.
pptxDoc.Save("SmartArt.pptx")
'Close the Presentation.
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Get the Slide from Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the SmartArt from slide.
ISmartArt smartArt = slide.Shapes[0] as ISmartArt;
//Remove a node at the specified index.
smartArt.Nodes.RemoveAt(4);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "SmartArt";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get the first slide from the Presentation.
ISlide slide = pptxDoc.Slides[0];
//Get the SmartArt from slide.
ISmartArt smartArt = slide.Shapes[0] as ISmartArt;
//Remove a node at the specified index.
smartArt.Nodes.RemoveAt(4);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("SmartArt.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the Presentation.
pptxDoc.Close();

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
// Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
// Get the first slide from the Presentation.
ISlide slide = pptxDoc.Slides[0];
// Get the SmartArt from slide.
ISmartArt smartArt = slide.Shapes[0] as ISmartArt;
// Remove a node at the specified index.
smartArt.Nodes.RemoveAt(4);
// Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
// Save Presentation in stream format.
pptxDoc.Save(stream);
// Close the presentation
pptxDoc.Close();
stream.Position = 0;
// The operation in Save under Xamarin varies between Windows Phone, Android
// and iOS platforms. Please refer presentation/xamarin section for respective
// code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("SmartArt.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("SmartArt.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Assistant nodes in SmartArt

You can check whether a node is an assistant or not. Also you can change a node as assistant node or revert an assistant node to normal node. The following code example demonstrates making an assistant node as normal node.

C#

```

// Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
// Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
// Add a SmartArt to the slide at the specified size and position
ISmartArt smartArt =
slide.Shapes.AddSmartArt(SmartArtType.OrganizationChart, 0, 0, 640, 426.96);
// Traverse through all nodes of the SmartArt.
foreach (ISmartArtNode node in smartArt.Nodes)
{
// Check if the node is assistant or not.
if (node.IsAssistant)
// Set the assistant node to false.
node.IsAssistant = false;
}
// Save the Presentation.

```

```
pptxDoc.Save("Sample.pptx");
//Close the Presentation.
pptxDoc.Close();
```

VB.NET

```
'Create an instance of PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Add a blank slide to the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a SmartArt to the slide at the specified size and position
Dim smartArt As ISmartArt =
slide.Shapes.AddSmartArt(SmartArtType.OrganizationChart, 0, 0, 640, 426.96)
'Traverse through all nodes of the SmartArt.
For Each node As ISmartArtNode In smartArt.Nodes
'Check if the node is assistant or not.
If node.IsAssistant Then
'Set the assistant node to false.
node.IsAssistant = False
End If
Next
'Save the Presentation.
pptxDoc.Save("Sample.pptx")
'Close the Presentation.
pptxDoc.Close()
```

UWP

```
//Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a SmartArt to the slide at the specified size and position
ISmartArt smartArt =
slide.Shapes.AddSmartArt(SmartArtType.OrganizationChart, 0, 0, 640, 426.96);
//Traverse through all nodes of the SmartArt.
foreach (ISmartArtNode node in smartArt.Nodes)
{
//Check if the node is assistant or not.
if (node.IsAssistant)
//Set the assistant node to false.
node.IsAssistant = false;
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "SmartArt";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
// Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a SmartArt to the slide at the specified size and position
ISmartArt smartArt =
slide.Shapes.AddSmartArt(SmartArtType.OrganizationChart, 0, 0, 640, 426.96);
//Traverse through all nodes of the SmartArt.
foreach (ISmartArtNode node in smartArt.Nodes)
{
    //Check if the node is assistant or not.
    if (node.IsAssistant)
    //Set the assistant node to false.
    node.IsAssistant = false;
}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("SmartArt.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the Presentation.
pptxDoc.Close();
```

XAMARIN

```
// Create an instance of PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a blank slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a SmartArt to the slide at the specified size and position
ISmartArt smartArt =
slide.Shapes.AddSmartArt(SmartArtType.OrganizationChart, 0, 0, 640, 426.96);
//Traverse through all nodes of the SmartArt.
foreach (ISmartArtNode node in smartArt.Nodes)
{
    //Check if the node is assistant or not.
    if (node.IsAssistant)
    //Set the assistant node to false.
    node.IsAssistant = false;
}
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("SmartArt.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("SmartArt.pptx",  
"application/vnd.openxmlformats-officedocument.presentationml.presentation",  
stream);
```

Limitations

The modifications in a SmartArt (like add/remove nodes, modify position and size of nodes etc., which involve SmartArt layout changes) done by Essential Presentation will not be reflected in Image and PDF conversion. Whereas layout changes will be reflected properly in the generated PPTX file when opened using Microsoft PowerPoint.

Supported SmartArt layout types

1. Basic Block List
2. Alternating Hexagons
3. Picture Caption List
4. Lined List
5. Vertical Bullet List
6. Vertical Box List
7. Horizontal Bullet List
8. Square Accent List
9. Picture Accent List
10. Bending Picture Accent List
11. Stacked List
12. Increasing Circle Process
13. Pie Process
14. Detailed Process
15. Grouped List
16. Horizontal Picture List
17. Continuous Picture List
18. Picture Strips
19. Vertical Picture List
20. Alternating Picture Blocks
21. Vertical Picture Accent List
22. Titled Picture Accent List
23. Vertical Block List
24. Vertical Chevron List
25. Vertical Accent List
26. Vertical Arrow List
27. Trapezoid List
28. Descending Block List
29. Table List
30. Segmented Process
31. Vertical Curved List
32. Pyramid List
33. Target List
34. Vertical Circle List
35. Table Hierarchy
36. Basic Process

37. Step Up Process
38. Step Down Process
39. Accent Process
40. Alternating Flow
41. Continuous Block Process
42. Increasing Arrows Process
43. Continuous Arrow Process
44. Process Arrows
45. Circle Accent Time Line
46. Basic Time Line
47. Basic Chevron Process
48. Closed Chevron Process
49. Chevron List
50. Sub-Step Process
51. Phased Process
52. Random to Result Process
53. Staggered Process
54. Process List
55. Circle Arrow Process
56. Basic Bending Process
57. Vertical Bending Process
58. Ascending Picture Accent Process
59. Upward Arrow
60. Descending Process
61. Circular Bending Process
62. Equation
63. Vertical Equation
64. Funnel
65. Gear
66. Arrow Ribbon
67. Opposing Arrows
68. Converging Arrows
69. Diverging Arrows
70. Basic Cycle
71. Text Cycle
72. Block Cycle
73. Non directional Cycle
74. Continuous Cycle
75. Multi Directional Cycle
76. Segmented Cycle
77. Basic Pie
78. Radial Cycle
79. Basic Radial
80. Diverging Radial
81. Radial Venn
82. Radial Cluster
83. Organization Chart
84. Name and Title Organization Chart

85. Half Circle Organization Chart
86. Circle Picture hierarchy
87. Hierarchy
88. Labeled Hierarchy
89. Horizontal Organization Chart
90. Horizontal Multi-Level Hierarchy
91. Horizontal Hierarchy
92. Horizontal Labeled Hierarchy
93. Balance
94. Circle Relationship
95. Hexagon Cluster
96. Opposing Ideas
97. Plus and Minus
98. Reverse List
99. Counter Balance Arrows
100. Segmented Pyramid
101. Nested Target
102. Converging Radial
103. Radial List
104. Basic Target
105. Basic Matrix
106. Titled Matrix
107. Grid Matrix
108. Cycle Matrix
109. Accent Picture
110. Circular Picture Callout
111. Snapshot Picture List
112. Spiral Picture
113. Captioned Pictures
114. Bending Picture Caption
115. Bending Picture-Semi Transparent Text
116. Bending Picture Blocks
117. Bending Picture Caption List
118. Titled Picture Blocks
119. Picture Grid
120. Picture Accent Blocks
121. Alternating Picture Circles
122. Title Picture Lineup
123. Picture Lineup
124. Framed Text Picture
125. Bubble Picture List
126. Basic Pyramid
127. Inverted Pyramid
128. Basic Venn
129. Linear Venn
130. Stacked Venn
131. Hierarchy List
132. Picture Accent Process

- 133. Repeating Bending Process
- 134. Vertical Process

Working with Comments

A comment is a text note attached to a location on a slide. Each comment contains an unformatted text string, information about its author and the time it was added. In a PowerPoint slide, the comments and the reply comments are sequentially maintained in a single collection. The top most comment will have the index position 0 and the other comments and replies in that slide will have the incremental index positions.

Adding a comment

The following code example demonstrates how to add a comment in a slide.

C#

```
//Create a PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a comment to the slide
slide.Comments.Add(10, 10, "Author1", "A1", "Can we change the font size to 20?", DateTime.Now);
//Save the Presentation
pptxDoc.Save("Comment.pptx");
//Close the Presentation
pptxDoc.Close();
```

VB.NET

```
'Create a PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Add a slide to the Presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add a comment to the slide
slide.Comments.Add(10, 10, "Author1", "A1", "Can we change the font size to 20?", DateTime.Now)
'Save the Presentation
pptxDoc.Save("Comment.pptx")
'Close the Presentation
pptxDoc.Close()
```

UWP

```
//Create a PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a comment to the slide
slide.Comments.Add(10, 10, "Author1", "A1", "Can we change the font size to 20?", DateTime.Now);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Comment";
```



```
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Create a PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a comment to the slide
slide.Comments.Add(10, 10, "Author1", "A1", "Can we change the font size to 20?", DateTime.Now);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Comment.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the Presentation
pptxDoc.Close();
```

XAMARIN

```
//Create a PowerPoint Presentation
IPresentation pptxDoc = Presentation.Create();
//Add a slide to the Presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add a comment to the slide
slide.Comments.Add(10, 10, "Author1", "A1", "Can we change the font size to 20?", DateTime.Now);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Comment.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Comment.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
```

Replying to a comment

The following code example demonstrates how to reply to an existing comment in a slide.

C#

```

//Create a PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Get the slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the comment in the slide
IComment comment = slide.Comments[0] as IComment;
//Add reply to the comment
slide.Comments.Add("Author2", "A2", "Yes, we can we change the font size to
20", DateTime.Now, comment);
//Save the presentation
pptxDoc.Save("ReplyComment.pptx");
//Close the Presentation
pptxDoc.Close();

```

VB.NET

```

'Create a PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Get the slide from the Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Get the comment in the slide
Dim comment As IComment = TryCast(slide.Comments(0), IComment)
'Add reply to the comment
slide.Comments.Add("Author2", "A2", "Yes, we can we change the font size to
20", DateTime.Now, comment)
'Save the presentation
pptxDoc.Save("ReplyComment.pptx")
'Close the Presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Get the slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the comment in the slide
IComment comment = slide.Comments[0] as IComment;
//Add reply to the comment
slide.Comments.Add("Author2", "A2", "Yes, we can we change the font size to
20", DateTime.Now, comment);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ReplyComment";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file

```

```
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get the slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the comment in the slide
IComment comment = slide.Comments[0] as IComment;
//Add reply to the comment
slide.Comments.Add("Author2", "A2", "Yes, we can we change the font size to 20", DateTime.Now, comment);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("ReplyComment.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Close the Presentation
pptxDoc.Close();
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream("Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get the slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the comment in the slide
IComment comment = slide.Comments[0] as IComment;
//Add reply to the comment
slide.Comments.Add("Author2", "A2", "Yes, we can we change the font size to 20", DateTime.Now, comment);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("ReplyComment.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("ReplyComment.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
```

Modifying the comment

The following code example demonstrates how to modify the content of a comment.

C#

```
//Create a PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Open a slide to the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the comment from the slide
IComment comment = slide.Comments[0] as IComment;
//Modify the comment text
comment.Text = "The comment text content is changed";
//Save the presentation
pptxDoc.Save("ModifyCommentText.pptx");
//Close the Presentation
pptxDoc.Close();
```

VB.NET

```
'Create a PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Open a slide to the Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Get the comment from the slide
Dim comment As IComment = TryCast(slide.Comments(0), IComment)
'Modify the comment text
comment.Text = "The comment text content is changed"
'Save the presentation
pptxDoc.Save("ModifyCommentText.pptx")
'Close the Presentation
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = await Presentation.OpenAsync(inputStorageFile);
//Get the slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the comment in the slide
IComment comment = slide.Comments[0] as IComment;
//Modify the comment text
comment.Text = "The comment text content is changed";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ModifyCommentText";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
```

```
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Open a slide to the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the comment from the slide
IComment comment = slide.Comments[0] as IComment;
//Modify the comment text
comment.Text = "The comment text content is changed";
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("ModifyCommentText.pptx",
FileMode.Create);
pptxDoc.Save(outputStream);
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream("Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Open a slide to the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the comment from the slide
IComment comment = slide.Comments[0] as IComment;
//Modify the comment text
comment.Text = "The comment text content is changed";
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("ModifyComment
Text.pptx", "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("ModifyCommentText.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

The following code example demonstrates how to modify the author name of a comment.

C#

```

//Create a PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Open a slide to the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the comment from the slide
IComment comment = slide.Comments[0] as IComment;
//Modify the comment text
comment.AuthorName = "NewAuthor";
//Save the presentation
pptxDoc.Save("ModifyCommentAuthor.pptx");
//Close the Presentation
pptxDoc.Close();

```

VB.NET

```

'Create a PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Open a slide to the Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Get the comment from the slide
Dim comment As IComment = TryCast(slide.Comments(0), IComment)
'Modify the comment text
comment.AuthorName = "NewAuthor"
'Save the presentation
pptxDoc.Save("ModifyCommentAuthor.pptx")
'Close the Presentation
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Get the slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the comment in the slide
IComment comment = slide.Comments[0] as IComment;
//Modify the comment author name
comment.AuthorName = "NewAuthor";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ModifyCommentAuthor";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Open a slide to the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the comment from the slide
IComment comment = slide.Comments[0] as IComment;
//Modify the comment text
comment.AuthorName = "NewAuthor";
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("ModifyCommentAuthor.pptx",
FileMode.Create);
pptxDoc.Save(outputStream);
//Close the Presentation
pptxDoc.Close();
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream("Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Open a slide to the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the comment from the slide
IComment comment = slide.Comments[0] as IComment;
//Modify the comment text
comment.AuthorName = "NewAuthor";
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("ModifyComment
Author.pptx", "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("ModifyCommentAuthor.pptx"
, "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
```

Deleting the comment

Deleting a comment will remove all its replies from the PowerPoint slide. You can also delete a particular reply comment from a slide. You can delete a comment by specifying its reference or by specifying its position.

The following code example demonstrates how to delete a comment from a slide.

C#

```
//Open a PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Get the first slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get a comment from the slide
IComment comment = slide.Comments[0];
//Remove the comment from the slide
slide.Comments.Remove(comment);
//Save the presentation
pptxDoc.Save("DeleteComment.pptx");
//Close the Presentation
pptxDoc.Close();
```

VB.NET

```
'Open a PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Get the first slide from the Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Get a comment from the slide
Dim comment As IComment = slide.Comments(0)
'Remove the comment from the slide
slide.Comments.Remove(comment)
'Save the presentation
pptxDoc.Save("DeleteComment.pptx")
'Close the Presentation
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Get the slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get the comment in the slide
IComment comment = slide.Comments[0] as IComment;
//Remove the comment from the slide
slide.Comments.Remove(comment);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
```



```

savePicker.SuggestedFileName = "DeleteComment";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get the first slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get a comment from the slide
IComment comment = slide.Comments[0];
//Remove the comment from the slide
slide.Comments.Remove(comment);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("DeleteComment.pptx",
    FileMode.Create);
pptxDoc.Save(outputStream);
//Close the Presentation
pptxDoc.Close();

```

XAMARIN

```

/"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream("Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get the first slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Get a comment from the slide
IComment comment = slide.Comments[0];
//Remove the comment from the slide
slide.Comments.Remove(comment);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DeleteComment
        .pptx", "application/vnd.openxmlformats-
        officedocument.presentationml.presentation", stream);
else

```

```
Xamarin.Forms.DependencyService.Get<ISave>().Save("DeleteComment.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

The following code example demonstrates how to delete a comment by specifying its position.

C#

```
//Open a PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Get the first slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Remove the first reply comment from the slide
slide.Comments.RemoveAt(1);
//Save the presentation
pptxDoc.Save("DeleteReplyComment.pptx");
//Close the Presentation
pptxDoc.Close();
```

VB.NET

```
'Open a PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Get the first slide from the Presentation
Dim slide As ISlide = pptxDoc.Slides(0)
'Remove the first reply comment from the slide
slide.Comments.RemoveAt(1)
'Save the presentation
pptxDoc.Save("DeleteReplyComment.pptx")
'Close the Presentation
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Get the slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Remove the first reply comment from the slide
slide.Comments.RemoveAt(1);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "DeleteReplyComment";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
```

```
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get the first slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Remove the first reply comment from the slide
slide.Comments.RemoveAt(1);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("DeleteReplyComment.pptx",
FileMode.Create);
pptxDoc.Save(outputStream);
//Close the Presentation
pptxDoc.Close();
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream("Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Get the first slide from the Presentation
ISlide slide = pptxDoc.Slides[0];
//Remove the first reply comment from the slide
slide.Comments.RemoveAt(1);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("DeleteReplyCo
mment.pptx", "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("DeleteReplyComment.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Working with Sections

Sections helps to manage the slides of a PowerPoint presentation. If a presentation has many slides, you can organize the slides using sections to make the navigation easier.

Creating a section

Adding a new slide to a section

The following code example demonstrates how to add a blank slide to a section.

C#

```
//Creates a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a section to the PowerPoint presentation
ISection section = pptxDoc.Sections.Add();
//Sets a name to the created section
section.Name = "SectionDemo";
//Adds a slide to the created section
ISlide slide = section.AddSlide(SlideLayoutType.Blank);
//Adds a text box to the slide
slide.AddTextBox(10, 10, 100, 100).TextBody.AddParagraph("Slide in
SectionDemo");
//Saves the PowerPoint presentation
pptxDoc.Save("Section.pptx");
```

VB.NET

```
'Creates a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Create()
'Adds a section to the PowerPoint presentation
Dim section As ISection = pptxDoc.Sections.Add()
'Sets a name to the created section
section.Name = "SectionDemo"
'Adds a slide to the created section
Dim slide As ISlide = section.AddSlide(SlideLayoutType.Blank)
'Adds a text box to the slide
slide.AddTextBox(10, 10, 100, 100).TextBody.AddParagraph("Slide in
SectionDemo")
'Saves the PowerPoint presentation
pptxDoc.Save("Section.PPTX")
```

UWP

```
//Creates a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a section to the PowerPoint presentation
ISection section = pptxDoc.Sections.Add();
//Sets a name to the created section
section.Name = "SectionDemo";
//Adds a slide to the created section
ISlide slide = section.AddSlide(SlideLayoutType.Blank);
//Adds a text box to the slide
slide.AddTextBox(10, 10, 100, 100).TextBody.AddParagraph("Slide in
SectionDemo");
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Section";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
```

```
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Creates a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a section to the PowerPoint presentation
ISection section = pptxDoc.Sections.Add();
//Sets a name to the created section
section.Name = "SectionDemo";
//Adds a slide to the created section
ISlide slide = section.AddSlide(SlideLayoutType.Blank);
//Adds a text box to the slide
slide.AddTextBox(10, 10, 100, 100).TextBody.AddParagraph("Slide in SectionDemo");
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Section.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
```

XAMARIN

```
//Creates a PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Adds a section to the PowerPoint presentation
ISection section = pptxDoc.Sections.Add();
//Sets a name to the created section
section.Name = "SectionDemo";
//Adds a slide to the created section
ISlide slide = section.AddSlide(SlideLayoutType.Blank);
//Adds a text box to the slide
slide.AddTextBox(10, 10, 100, 100).TextBody.AddParagraph("Slide in SectionDemo");
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Section.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Section.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Adding an existing slide to a section

The following code example demonstrates how to add an existing slide to a section.

C#

```
//Loads a PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("PPTXWithoutSection.PPTX");
//Creates a new section in the PowerPoint presentation
pptxDoc.Sections.Add();
//Moves the first slide to the created section
pptxDoc.Slides[0].MoveToSection(0);
//Saves the PowerPoint presentation
pptxDoc.Save("Sections.PPTX");
```

VB.NET

```
'Loads a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("PPTXWithoutSection.PPTX")
'Creates a new section in the PowerPoint presentation
pptxDoc.Sections.Add()
'Moves the first slide to the created section
pptxDoc.Slides(0).MoveToSection(0)
'Saves the PowerPoint presentation
pptxDoc.Save("Sections.PPTX")
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Creates a new section in the PowerPoint presentation
pptxDoc.Sections.Add();
//Moves the first slide to the created section
pptxDoc.Slides[0].MoveToSection(0);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sections";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new
FileStream("PPTXWithSections.PPTX", FileMode.Open);
```

```

IPresentation pptxDoc = Presentation.Open(inputStream);
//Creates a new section in the PowerPoint presentation
pptxDoc.Sections.Add();
//Moves the first slide to the created section
pptxDoc.Slides[0].MoveToSection(0);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Section.pptx", FileMode.Create);
pptxDoc.Save(outputStream);

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Creates a new section in the PowerPoint presentation
pptxDoc.Sections.Add();
//Moves the first slide to the created section
pptxDoc.Slides[0].MoveToSection(0);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Section.pptx",
, "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Section.pptx.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Inserting a section

The following code example demonstrates how to insert a section in a template PowerPoint presentation that contains sections

C#

```

//Loads a PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("PPTXWithSections.PPTX");
//Creates a new section
ISection section = pptxDoc.Sections.Add();
//Names the created section
section.Name = "InsertedSection";
//Inserts the section at second position.

```

```
pptxDoc.Sections.Insert(1, section);
//Saves the PowerPoint presentation
pptxDoc.Save("Sections.PPTX");
```

VB.NET

```
'Loads a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("PPTXWithSections.PPTX")
'Creates a new section
Dim section As ISection = pptxDoc.Sections.Add()
'Names the created section
section.Name = "InsertedSection"
'Inserts the section at second position.
pptxDoc.Sections.Insert(1, section)
'Saves the PowerPoint presentation
pptxDoc.Save("Sections.PPTX")
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
/Creates a new section
ISection section = pptxDoc.Sections.Add();
//Names the created section
section.Name = "InsertedSection";
//Inserts the section at second position.
pptxDoc.Sections.Insert(1, section);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sections";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new
FileStream("PPTXWithSections.PPTX", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Creates a new section
ISection section = pptxDoc.Sections.Add();
//Names the created section
section.Name = "InsertedSection";
//Inserts the section at second position.
```



```
pptxDoc.Sections.Insert(1, section);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Section.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
```

XAMARIN

```
/"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Creates a new section
ISection section = pptxDoc.Sections.Add();
//Names the created section
section.Name = "InsertedSection";
//Inserts the section at second position.
pptxDoc.Sections.Insert(1, section);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Section.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Section.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
```

Moving the sections within a PowerPoint presentation

You can move the sections within a PowerPoint presentation. The following code example demonstrates how to move a section to a specific position in the navigation pane.

C#

```
//Loads a PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("PPTXWithSections.PPTX");
//Moves the second section to third position within the PowerPoint presentation.
pptxDoc.Sections[2].Move(3);
//Saves the PowerPoint presentation
pptxDoc.Save("Sections.PPTX");
```

VB.NET

```
'Loads a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("PPTXWithSections.PPTX")
'Moves the second section to third position within the PowerPoint presentation.
pptxDoc.Sections(2).Move(3)
'Saves the PowerPoint presentation
pptxDoc.Save("Sections.PPTX")
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Moves the second section to third position within the PowerPoint presentation.
pptxDoc.Sections[2].Move(3);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sections";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new
FileStream("PPTXWithSections.PPTX", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Moves the second section to third position within the PowerPoint presentation.
pptxDoc.Sections[2].Move(3);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Section.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
```

XAMARIN

```
/"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
```

```

IPresentation pptxDoc = Presentation.Open(inputStream);
//Moves the second section to third position within the PowerPoint
presentation.
pptxDoc.Sections[2].Move(3);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Section.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Section.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Moving a slide within sections

The following code example demonstrates how to move a slide from one section to another.

C#

```

//Loads a PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("PPTXWithSections.PPTX");
//Gets the first slide of second section in the PowerPoint presentation
ISlide slide = pptxDoc.Sections[1].Slides[0];
//Moves the slide to first section
slide.MoveToSection(0);
//Saves the PowerPoint presentation
pptxDoc.Save("Sections.PPTX");

```

VB.NET

```

'Loads a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("PPTXWithSections.PPTX")
'Gets the first slide of second section in the PowerPoint presentation
Dim slide As ISlide = pptxDoc.Sections(1).Slides(0)
'Moves the slide to first section
slide.MoveToSection(0)
'Saves the PowerPoint presentation
pptxDoc.Save("Sections.PPTX")

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");

```

```

//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Gets the first slide of second section in the PowerPoint presentation
ISlide slide = pptxDoc.Sections[1].Slides[0];
//Moves the slide to first section
slide.MoveToSection(0);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sections";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new
FileStream("PPTXWithSections.PPTX", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide of second section in the PowerPoint presentation
ISlide slide = pptxDoc.Sections[1].Slides[0];
//Moves the slide to first section
slide.MoveToSection(0);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Section.pptx", FileMode.Create);
pptxDoc.Save(outputStream);

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets the first slide of second section in the PowerPoint presentation
ISlide slide = pptxDoc.Sections[1].Slides[0];
//Moves the slide to first section
slide.MoveToSection(0);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Section.pptx"
, "application/vnd.openxmlformats-
officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Section.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Cloning and merging the slides in a section

The following code example demonstrates how to clone the slide collection of a section and add those slides to a destination presentation.

C#

```

//Loads a PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("PPTXWithSections.PPTX");
//Clones the slides in 3rd section
ISlides slides = pptxDoc.Sections[2].Clone();
//Creates a destination PowerPoint presentation instance. Existing
presentations can also be used here.
pptxDoc = Presentation.Create();
//Iterates the cloned slides and adds the slides to the destination
presentation
foreach (ISlide slide in slides)
pptxDoc.Slides.Add(slide);
//Saves the PowerPoint presentation
pptxDoc.Save("Sections.PPTX");

```

VB.NET

```

'Loads a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("PPTXWithSections.PPTX")
'Clones the slides in 3rd section
Dim slides As ISlides = pptxDoc.Sections(2).Clone()
'Creates a destination PowerPoint presentation instance. Existing
presentations can also be used here.
pptxDoc = Presentation.Create()
'Iterates the cloned slides and adds the slides to the destination
presentation
For Each slide As ISlide In slides
pptxDoc.Slides.Add(slide)
Next
'Save the PowerPoint presentation
pptxDoc.Save("Sections.PPTX")

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();

```

```

//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Clones the slides in 3rd section
ISlides slides = pptxDoc.Sections[2].Clone();
//Creates a destination PowerPoint presentation instance. Existing presentations can also be used here.
pptxDoc = Presentation.Create();
//Iterates the cloned slides and adds the slides to the destination presentation
foreach (ISlide slide in slides)
pptxDoc.Slides.Add(slide);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sections";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new
FileStream("PPTXWithSections.PPTX", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Clones the slides in 3rd section
ISlides slides = pptxDoc.Sections[2].Clone();
//Creates a destination PowerPoint presentation instance. Existing presentations can also be used here.
pptxDoc = Presentation.Create();
//Iterates the cloned slides and adds the slides to the destination presentation
foreach (ISlide slide in slides)
pptxDoc.Slides.Add(slide);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Section.pptx", FileMode.Create);
pptxDoc.Save(outputStream);

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Clones the slides in 3rd section
ISlides slides = pptxDoc.Sections[2].Clone();
//Creates a destination PowerPoint presentation instance. Existing presentations can also be used here.
pptxDoc = Presentation.Create();

```

```

//Iterates the cloned slides and adds the slides to the destination presentation
foreach (ISlide slide in slides)
    pptxDoc.Slides.Add(slide);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Section.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
    Xamarin.Forms.DependencyService.Get<ISave>().Save("Section.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);

```

Removing a section

The following code example demonstrates how to create remove a particular section from the sections collection of a presentation.

C#

```

//Loads a PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("PPTXWithSections.PPTX");
//Removes the second section from the PowerPoint presentation
pptxDoc.Sections.Remove(pptxDoc.Sections[1]);
//Saves the PowerPoint presentation
pptxDoc.Save("Sections.PPTX");

```

VB.NET

```

'Loads a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("PPTXWithSections.PPTX")
'Removes the second section from the PowerPoint presentation
pptxDoc.Sections.Remove(pptxDoc.Sections(1))
'Saves the PowerPoint presentation
pptxDoc.Save("Sections.PPTX")

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation

```

```

IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Removes the second section from the PowerPoint presentation
pptxDoc.Sections.Remove(pptxDoc.Sections[1]);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sections";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new
FileStream("PPTXWithSections.PPTX", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Removes the second section from the PowerPoint presentation
pptxDoc.Sections.Remove(pptxDoc.Sections[1]);
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Section.pptx", FileMode.Create);
pptxDoc.Save(outputStream);

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Removes the second section from the PowerPoint presentation
pptxDoc.Sections.Remove(pptxDoc.Sections[1]);
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Section.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Section.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```


Remove all sections

The following code example demonstrates how to remove section collection from an existing PowerPoint presentation.

C#

```
//Loads a PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("PPTXWithSections.PPTX");
//Removes the sections
pptxDoc.Sections.Clear();
//Saves the PowerPoint presentation
pptxDoc.Save("Sections.PPTX");
```

VB.NET

```
'Loads a PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("PPTXWithSections.PPTX")
'Removes the sections
pptxDoc.Sections.Clear()
'Saves the PowerPoint presentation
pptxDoc.Save("Sections.PPTX")
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Removes the sections
pptxDoc.Sections.Clear();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sections";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Loads or open an PowerPoint Presentation
FileStream inputStream = new
FileStream("PPTXWithSections.PPTX", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Removes the sections
pptxDoc.Sections.Clear();
```

```
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Sections.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
```

XAMARIN

```
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Removes the sections
pptxDoc.Sections.Clear();
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer presentation/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Sections.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Sections.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
```

Working with Macros

A macro is a series of commands that can be grouped together as a single command to automate a frequently used tasks. Macros can be created for Microsoft PowerPoint using Visual Basic for Applications (VBA). Please refer [Create Macros in PowerPoint](#) for more details

Macros enabled PowerPoint presentations of file types - .PPTM and .POTM can only be preserved using Essential Presentation.

Loading and Saving Macro enabled Presentation

The following code illustrates how to load and save a macro enabled presentation.

C#

```
//Opens an existing macro enabled PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Sample.PPTM");
//Adds a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a text box to the slide
IParagraph paragraph = slide.Shapes.AddTextBox(100, 100, 300, 80).TextBody.AddParagraph("Preserve Macros");
```

```
//Saves the presentation
pptxDoc.Save("Output.PPTM");
//Closes the presentation
pptxDoc.Close();
```

VB.NET

```
'Opens an existing macro enabled PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.PPTM")
'Adds a blank slide to the presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Adds a text box to the slide
Dim paragraph As IParagraph = slide.Shapes.AddTextBox(100, 100, 300,
80).TextBody.AddParagraph("Preserve Macros")
'Saves the presentation
pptxDoc.Save("Output.PPTM")
'Closes the presentation
pptxDoc.Close()
```

UWP

```
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".PPTM");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Adds a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a text box to the slide
IParagraph paragraph = slide.Shapes.AddTextBox(100, 100, 300,
80).TextBody.AddParagraph("Preserve Macros");
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".PPTM" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Opens an existing macro enabled PowerPoint presentation
FileStream inputStream = new FileStream("Sample.PPTM", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Adds a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a text box to the slide
IParagraph paragraph = slide.Shapes.AddTextBox(100, 100, 300,
80).TextBody.AddParagraph("Preserve Macros");
```

```
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.PPTM", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the presentation
pptxDoc.Close();
```

XAMARIN

```
//Opens an existing macro enabled PowerPoint presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Adds a blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Adds a text box to the slide
IParagraph paragraph = slide.Shapes.AddTextBox(100, 100, 300,
80).TextBody.AddParagraph("Preserve Macros");
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.PPTM",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.PPTM",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Removing Macros from Macro enabled Presentation

The following code example illustrates how to remove the macros present in the presentation,

C#

```
//Opens an existing macro enabled PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Sample.PPTM");
//Checks whether the presentation has macros and then removes them
if (pptxDoc.HasMacros)
pptxDoc.RemoveMacros();
//Saves the presentation
pptxDoc.Save("Output.pptx");
//Closes the presentation
pptxDoc.Close();
```

VB.NET

```
'Opens an existing macro enabled PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.PPTM")
'Checks whether the presentation has macros and then removes them
```

```

If pptxDoc.HasMacros Then
pptxDoc.RemoveMacros ()
End If
'Saves the presentation
pptxDoc.Save ("Output.pptx")
'Closes the presentation
pptxDoc.Close ()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker ();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add (".PPTM");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync ();
//Loads or open an Macro enabled PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync (inputStorageFile);
//Checks whether the presentation has macros and then removes them
if (pptxDoc.HasMacros)
pptxDoc.RemoveMacros ();
//Saves the presentation
pptxDoc.Save ("Output.pptx");
//Closes the presentation
pptxDoc.Close ();

```

ASP.NET CORE

```

//Opens an existing macro enabled PowerPoint presentation
FileStream inputStream = new FileStream ("Sample.PPTM", FileMode.Open);
IPresentation pptxDoc = Presentation.Open (inputStream);
//Checks whether the presentation has macros and then removes them
if (pptxDoc.HasMacros)
pptxDoc.RemoveMacros ();
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream ("Output.pptx", FileMode.Create);
pptxDoc.Save (outputStream);
//Closes the presentation
pptxDoc.Close ();

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof (App).GetTypeInfo ().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream ("SampleBrowser.Presentation.Samples.Template.Sample.PPTM");
//Opens an existing macro enabled PowerPoint presentation
IPresentation pptxDoc = Presentation.Open (inputStream);
//Checks whether the presentation has macros and then removes them
if (pptxDoc.HasMacros)
pptxDoc.RemoveMacros ();
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream ();
//Save Presentation in stream format.

```

```
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
```

Security

Encrypting with password

You can protect a PowerPoint Presentation by encrypting the document by using a password. This prevents unauthorized users to access or make changes in the Presentation.

The following code example demonstrates how to encrypt a PowerPoint Presentation with password.

C#

```
//Creates an instance for Presentation
IPresentation presentation = Presentation.Create();
//Adds slide to Presentation
ISlide slide = presentation.Slides.Add(SlideLayoutType.Blank);
//Adds textbox to slide
IShape shape = slide.Shapes.AddTextBox(100, 30, 200, 300);
//Adds a paragraph with text content.
IParagraph paragraph = shape.TextBody.AddParagraph("Password Protected.");
//Protects the file with password
presentation.Encrypt("PASSWORD!@1#$");
//Saves the Presentation
presentation.Save("Sample.pptx");
//Closes the Presentation
presentation.Close();
```

VB.NET

```
'Creates an instance for Presentation
Dim presentationDocument As IPresentation = Presentation.Create()
'Adds slide to Presentation
Dim slide As ISlide = presentationDocument.Slides.Add(SlideLayoutType.Blank)
'Adds textbox to slide
Dim shape As IShape = slide.Shapes.AddTextBox(100, 30, 200, 300)
'Adds a paragraph with text content.
Dim paragraph As IParagraph = shape.TextBody.AddParagraph("Password
Protected.")
'Protects the file with password
presentationDocument.Encrypt("PASSWORD!@1#$")
'Saves the Presentation
```

```
presentationDocument.Save("Sample.pptx")  
'Closes the Presentation  
presentationDocument.Close()
```

Note: PowerPoint Presentation doesn't support encryption in ASP.NET Core, Blazor and Xamarin.

Decrypting the PowerPoint Presentation

Essential Presentation provides ability to remove the encryption from the PowerPoint Presentation. You can decrypt a PowerPoint Presentation by opening it with the password.

Opening the Encrypted PowerPoint Presentation

The following code example demonstrates opening the encrypted PowerPoint Presentation.

C#

```
//Opens an existing Presentation from file system and it can be decrypted by  
using the provided password.  
IPresentation presentation = Presentation.Open("Sample.pptx",  
"PASSWORD!@1#$");  
//Saves the Presentation  
presentation.Save("Output.pptx");  
//Closes the Presentation  
presentation.Close();
```

VB.NET

```
'Opens an existing Presentation from file system and it can be decrypted by  
using the provided password.  
Dim presentationDocument As IPresentation = Presentation.Open("Sample.pptx",  
"PASSWORD!@1#$")  
'Saves the Presentation  
presentationDocument.Save("Output.pptx")  
'Closes the Presentation  
presentationDocument.Close()
```

Removing the encryption from Presentation

The following code example demonstrates removing the encryption from a PowerPoint Presentation.

C#

```
//Opens an existing Presentation from file system and it can be decrypted by  
using the provided password.  
IPresentation presentation = Presentation.Open("Sample.pptx",  
"PASSWORD!@1#$");  
//Decrypts the document  
presentation.RemoveEncryption();  
//Saves the presentation  
presentation.Save("Output.pptx");  
//Closes the Presentation  
presentation.Close();
```

VB.NET

```
'Opens an existing Presentation from file system and it can be decrypted by
using the provided password.
Dim presentationDocument As IPresentation = Presentation.Open("Sample.pptx",
"PASSWORD!@1#$")
'Decrypts the document
presentationDocument.RemoveEncryption()
'Saves the Presentation
presentationDocument.Save("Output.pptx")
'Closes the Presentation
presentationDocument.Close()
```

Write Protection

You can set write protection for a PowerPoint Presentation and remove protection from the write protected PowerPoint presentation.

Protect PowerPoint Presentation

You can protect a PowerPoint Presentation with password to restrict unauthorized editing.

The following code example shows how to set write protection for a PowerPoint Presentation.

C#

```
//Create a new instance for PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add the blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add the shape to the slide
IShape shape = slide.Shapes.AddShape(AutoShapeType.BlockArc, 0, 0, 200,
200);
//Add the paragraph to the shape.
IParagraph paragraph = shape.TextBody.AddParagraph("welcome");
//Sets the author name
pptxDoc.BuiltInDocumentProperties.Author = "Syncfusion";
//Set the write protection for presentation instance
pptxDoc.SetWriteProtection("MYPASSWORD");
//Saves the modified cloned PowerPoint presentation
pptxDoc.Save("Sample.pptx");
//Close the presentation instance
pptxDoc.Close();
```

VB.NET

```
'Create a new instance for PowerPoint presentation
IPresentation pptxDoc = Presentation.Create()
'Add the blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Add the shape to the slide
IShape shape = slide.Shapes.AddShape(AutoShapeType.BlockArc, 0, 0, 200, 200)
'Add the paragraph to the shape.
IParagraph paragraph = shape.TextBody.AddParagraph("welcome")
'Sets the author name
pptxDoc.BuiltInDocumentProperties.Author = "Syncfusion"
'Set the write protection for presentation instance
pptxDoc.SetWriteProtection("MYPASSWORD")
'Saves the modified cloned PowerPoint presentation
pptxDoc.Save("Sample.pptx")
```



```
'Close the presentation instance
pptxDoc.Close()
```

UWP

```
//Create a new instance for PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add the blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add the shape to the slide
IShape shape = slide.Shapes.AddShape(AutoShapeType.BlockArc, 0, 0, 200,
200);
//Add the paragraph to the shape.
IParagraph paragraph = shape.TextBody.AddParagraph("welcome");
//Sets the author name
pptxDoc.BuiltInDocumentProperties.Author = "Syncfusion";
//Set the write protection for presentation instance
pptxDoc.SetWriteProtection("MYPASSWORD");
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Create a new instance for PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add the blank slide to the presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add the shape to the slide
IShape shape = slide.Shapes.AddShape(AutoShapeType.BlockArc, 0, 0, 200,
200);
//Add the paragraph to the shape.
IParagraph paragraph = shape.TextBody.AddParagraph("welcome");
//Sets the author name
pptxDoc.BuiltInDocumentProperties.Author = "Syncfusion";
//Set the write protection for presentation instance
pptxDoc.SetWriteProtection("MYPASSWORD");
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the presentation
pptxDoc.Close();
```

XAMARIN

```
//Create a new instance for PowerPoint presentation
IPresentation pptxDoc = Presentation.Create();
//Add the blank slide to the presentation
```

```

ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Add the shape to the slide
IShape shape = slide.Shapes.AddShape(AutoShapeType.BlockArc, 0, 0, 200,
200);
//Add the paragraph to the shape.
IParagraph paragraph = shape.TextBody.AddParagraph("welcome");
//Sets the author name
pptxDoc.BuiltInDocumentProperties.Author = "Syncfusion";
//Set the write protection for presentation instance
pptxDoc.SetWriteProtection("MYPASSWORD");
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Remove Protection

You can check whether a PowerPoint Presentation is write protected and remove protection from the write protected PowerPoint Presentation.

The following code example shows how to remove restriction protection from the write protected PowerPoint Presentation

C#

```

//Open the PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Gets whether the presentation is write Protected. Read - only.
bool writeProtected = pptxDoc.IsWriteProtected;
//Checks whether the presentation is write protected
if (writeProtected)
{
//Removes the write protection for presentation instance.
pptxDoc.RemoveWriteProtection();
}
//Saves the modified cloned PowerPoint presentation
pptxDoc.Save("Output.pptx");
//Close the presentation instance
pptxDoc.Close();

```

VB.NET

```

'Open the PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
'Gets whether the presentation is write Protected. Read - only.
bool writeProtected = pptxDoc.IsWriteProtected;
'Checks whether the presentation is write protected
if (writeProtected)
{
    'Removes the write protection for presentation instance.
    pptxDoc.RemoveWriteProtection();
}
'Saves the modified cloned PowerPoint presentation
pptxDoc.Save("Output.pptx")
'Close the presentation instance
pptxDoc.Close()

```

UWP

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".pptx");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc= await Presentation.OpenAsync(inputStorageFile);
//Gets whether the presentation is write Protected. Read - only.
bool writeProtected = pptxDoc.IsWriteProtected;
//Checks whether the presentation is write protected
if (writeProtected)
{
    //Removes the write protection for presentation instance.
    pptxDoc.RemoveWriteProtection();
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
    ".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);

```

ASP.NET CORE

```

//Loads or open an PowerPoint Presentation
FileStream inputStream = new FileStream("Sample.pptx", FileMode.Open);
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets whether the presentation is write Protected. Read - only.
bool writeProtected = pptxDoc.IsWriteProtected;
//Checks whether the presentation is write protected
if (writeProtected)
{
    //Removes the write protection for presentation instance.
    pptxDoc.RemoveWriteProtection();
}

```

```

}
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("Output.pptx", FileMode.Create);
pptxDoc.Save(outputStream);
//Closes the presentation
pptxDoc.Close();

```

XAMARIN

```

//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.Presentation.Samples.Template.Sample.pptx");
//Loads or open an PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open(inputStream);
//Gets whether the presentation is write Protected. Read - only.
bool writeProtected = pptxDoc.IsWriteProtected;
//Checks whether the presentation is write protected
if (writeProtected)
{
//Removes the write protection for presentation instance.
pptxDoc.RemoveWriteProtection();
}
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("Output.pptx",
"application/vnd.openxmlformats-officedocument.presentationml.presentation",
stream);

```

Note: 1. In Xamarin application, this feature is supported from the target framework .NET Standard 2.0 version onwards.

2. For ASP.NET Core, this feature is supported from .NET Core 2.0 version onwards.

Working with OLE Objects

The OLE Object enables sharing of application objects written in different file formats. In PowerPoint presentation the application data can be inserted into a PowerPoint slide using the [programmatic identifier](#) of each file format.

Inserting OLE Object to a Slide

The below code snippet demonstrates how to add an Excel worksheet into a slide.

C#

```
//Create new instance of PowerPoint presentation. [Equivalent to launching
MS PowerPoint with no slides].
IPresentation pptxDoc = Presentation.Create();
//Add slide with blank layout to presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Get the excel file as stream
Stream excelStream = File.Open("OleTemplate.xlsx", FileMode.Open);
//Image to be displayed, This can be any image
Stream imageStream = File.Open("OlePicture.png", FileMode.Open);
//Add an OLE object to the slide
IOleObject oleObject = slide.Shapes.AddOleObject(imageStream,
"Excel.Sheet.12", excelStream);
//Set size and position of the OLE object
oleObject.Left = 10;
oleObject.Top = 10;
oleObject.Width = 400;
oleObject.Height = 300;
//Save the presentation
pptxDoc.Save("OleObjectSample.pptx");
//Close the presentation
pptxDoc.Close();
```

VB.NET

```
'Create new instance of PowerPoint presentation. [Equivalent to launching MS
PowerPoint with no slides].
Dim pptxDoc As IPresentation = Presentation.Create()
'Add slide with blank layout to presentation
Dim slide As ISlide = pptxDoc.Slides.Add(SlideLayoutType.Blank)
'Get the excel file as stream
Dim excelStream As Stream = File.Open("OleTemplate.xlsx", FileMode.Open)
'Image to be displayed, This can be any image
Dim imageStream As Stream = File.Open("OlePicture.png", FileMode.Open)
'Add an OLE object to the slide
Dim oleObject As IOleObject = slide.Shapes.AddOleObject(imageStream,
"Excel.Sheet.12", excelStream)
'Set size and position of the OLE object
oleObject.Left = 10
oleObject.Top = 10
oleObject.Width = 400
oleObject.Height = 300
'Save the presentation
pptxDoc.Save("OleObjectSample.pptx")
'Close the presentation
pptxDoc.Close()
```

UWP

```
//Create new instance of PowerPoint presentation. [Equivalent to launching
MS PowerPoint with no slides].
IPresentation pptxDoc = Presentation.Create();
```

```
//Add slide with blank layout to presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Get the excel file as stream
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream excelStream =
assembly.GetManifestResourceStream("UWP.Data.OleTemplate.xlsx");
//Image to be displayed, This can be any image
Stream imageStream =
assembly.GetManifestResourceStream("UWP.Data.OlePicture.png");
//Add an OLE object to the slide
IOleObject oleObject = slide.Shapes.AddOleObject(imageStream,
"Excel.Sheet.12", excelStream);
//Set size and position of the OLE object
oleObject.Left = 10;
oleObject.Top = 10;
oleObject.Width = 400;
oleObject.Height = 300;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "OleObjectSample";
savePicker.FileTypeChoices.Add("PowerPoint Files", new List<string>() {
".pptx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await pptxDoc.SaveAsync(storageFile);
```

ASP.NET CORE

```
//Create new instance of PowerPoint presentation.
IPresentation pptxDoc = Presentation.Create();
//Add slide with blank layout to presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//Get the excel file as stream
FileStream excelStream = new FileStream("OleTemplate.xlsx", FileMode.Open);
//Image to be displayed, This can be any image
FileStream excelStream = new FileStream("OlePicture.png", FileMode.Open);
//Add an OLE object to the slide
IOleObject oleObject = slide.Shapes.AddOleObject(imageStream,
"Excel.Sheet.12", excelStream);
//Set size and position of the OLE object
oleObject.Left = 10;
oleObject.Top = 10;
oleObject.Width = 400;
oleObject.Height = 300;
//Save the PowerPoint Presentation as stream
FileStream outputStream = new FileStream("OleObjectSample.pptx",
FileMode.Create);
pptxDoc.Save(outputStream);
//Close the presentation
pptxDoc.Close();
```

XAMARIN

```
//Create new instance of PowerPoint presentation.
```

```

IPresentation pptxDoc = Presentation.Create();
//Add slide with blank layout to presentation
ISlide slide = pptxDoc.Slides.Add(SlideLayoutType.Blank);
//"App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Get the excel file as stream
Stream excelStream = assembly.GetManifestResourceStream("OleTemplate.xlsx");
//Image to be displayed, This can be any image
Stream imageStream = assembly.GetManifestResourceStream("OlePicture.png");
//Add an OLE object to the slide
IOleObject oleObject = slide.Shapes.AddOleObject(imageStream,
"Excel.Sheet.12", excelStream);
//Set size and position of the OLE object
oleObject.Left = 10;
oleObject.Top = 10;
oleObject.Width = 400;
oleObject.Height = 300;
//Create new memory stream to save Presentation.
MemoryStream stream = new MemoryStream();
//Save Presentation in stream format.
pptxDoc.Save(stream);
//Close the presentation
pptxDoc.Close();
stream.Position = 0;
//The operation in Save under Xamarin varies between Windows Phone, Android
and iOS platforms. Please refer presentation/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("OleObjectSample.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);
else
Xamarin.Forms.DependencyService.Get<ISave>().Save("OleObjectSample.pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation", stream);

```

Presentation to image conversion

.NET Framework

This section covers converting an entire Presentation or a single slide to image in Windows Forms, WPF, ASP.NET and ASP.NET MVC platforms. The supported image formats are listed as follows.

- BMP
- EMF
- JPG
- JPEG
- PNG

Assemblies Required

Refer to the following links for assemblies required based on platforms to convert the worksheet to image.

- [Assemblies Information](#)
- [NuGet Information](#)

Tips: When converting a slide to image, use 'Metafile' format for good image resolution.

The following code example demonstrates how to convert a slide to image.

C#

```
//Namespaces to perform PPTX to Image conversion
using Syncfusion.Presentation;
using Syncfusion.OfficeChartToImageConverter;
using System.IO;
using Syncfusion.Drawing;
//Opens a PowerPoint Presentation file
IPresentation pptxDoc = Presentation.Open(fileName);
//Creates an instance of ChartToImageConverter
pptxDoc.ChartToImageConverter = new ChartToImageConverter();
//Sets the scaling mode as best
pptxDoc.ChartToImageConverter.ScalingMode =
Syncfusion.OfficeChart.ScalingMode.Best;
//Converts the first slide into image
Image image =
pptxDoc.Slides[0].ConvertToImage(Syncfusion.Drawing.ImageType.Metafile);
//Saves the image as file
image.Save("slide1.png");
//Disposes the image
image.Dispose();
//Closes the Presentation instance
pptxDoc.Close();
```

VB.NET

```
'Namespaces to perform PPTX to Image conversion
Imports Syncfusion.Presentation
Imports Syncfusion.OfficeChartToImageConverter
Imports Syncfusion.Drawing
Imports System.IO
'Opens a PowerPoint Presentation file
Dim pptxDoc As IPresentation = Presentation.Open(fileName)
'Creates an instance of ChartToImageConverter
pptxDoc.ChartToImageConverter = New ChartToImageConverter()
'Sets the scaling mode as best
pptxDoc.ChartToImageConverter.ScalingMode =
Syncfusion.OfficeChart.ScalingMode.Best
'Converts the first slide into image
Dim image As Image =
pptxDoc.Slides(0).ConvertToImage(Syncfusion.Drawing.ImageType.Metafile)
'Saves the image as file
image.Save("slide1.png")
'Disposes the image
image.Dispose()
'Closes the Presentation instance
Presentation 1.Close()
```


ASP.NET CORE

```
//Namespaces to perform PPTX to Image conversion
using Syncfusion.Presentation;
using Syncfusion.PresentationRenderer;
using System.IO;
//Open the existing PowerPoint presentation with stream.
using (IPresentation pptxDoc = Presentation.Open(fileStreamInput))
{
    //Initialize the PresentationRenderer to perform image conversion.
    pptxDoc.PresentationRenderer = new PresentationRenderer();
    //Convert PowerPoint slide to image as stream.
    using (Stream stream =
        pptxDoc.Slides[0].ConvertToImage(ExportImageFormat.Jpeg))
    {
        //Reset the stream position
        stream.Position = 0;
        //Create the output image file stream
        using (FileStream fileStreamOutput = File.Create("Output.jpg"))
        {
            //Copy the converted image stream into created output stream
            stream.CopyTo(fileStreamOutput);
        }
    }
}
```

XAMARIN

```
//Namespaces to perform PPTX to Image conversion
using Syncfusion.Presentation;
using Syncfusion.PresentationRenderer;
using System.IO;
//Open the existing PowerPoint presentation with stream.
using (IPresentation pptxDoc = Presentation.Open(fileStreamInput))
{
    //Initialize the PresentationRenderer to perform image conversion.
    pptxDoc.PresentationRenderer = new PresentationRenderer();
    //Convert PowerPoint slide to image as stream.
    using (Stream stream =
        pptxDoc.Slides[0].ConvertToImage(ExportImageFormat.Jpeg))
    {
        //Reset the stream position
        stream.Position = 0;
        //Create the output image file stream
        using (FileStream fileStreamOutput = File.Create("Output.jpg"))
        {
            //Copy the converted image stream into created output stream
            stream.CopyTo(fileStreamOutput);
        }
    }
}
```

The following code example demonstrates the conversion of an entire Presentation to images:

C#

```

//Loads the PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Creates instance of ChartToImageConverter
pptxDoc.ChartToImageConverter = new ChartToImageConverter();
//Sets the scaling mode as best
pptxDoc.ChartToImageConverter.ScalingMode =
Syncfusion.OfficeChart.ScalingMode.Best;
//Converts entire Presentation to images
Image[] images =
pptxDoc.RenderAsImages(Syncfusion.Drawing.ImageType.Metafile);
//Saves the image to file system
foreach (Image image in images)
{
    image.Save("ImageOutput" + Guid.NewGuid().ToString() + ".png");
}

```

VB.NET

```

'Loads the PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Creates instance of ChartToImageConverter
pptxDoc.ChartToImageConverter = New ChartToImageConverter()
'Sets the scaling mode as best
pptxDoc.ChartToImageConverter.ScalingMode =
Syncfusion.OfficeChart.ScalingMode.Best
'Converts entire Presentation to images
Dim images As Image() =
pptxDoc.RenderAsImages(Syncfusion.Drawing.ImageType.Metafile)
'Saves the image to file system
For Each image As Image In images
    image.Save("ImageOutput" + Guid.NewGuid().ToString() + ".png")
Next

```

The following code snippet demonstrates how to convert a PowerPoint slide to image using custom image resolution,

C#

```

//Loads the PowerPoint presentation
IPresentation pptxDoc = Presentation.Open("Output.pptx");
//Declare variables to hold custom width and height
int customWidth = 1500;
int customHeight = 1000;
//Converts the slide as image and returns the image stream
Stream stream =
pptxDoc.Slides[0].ConvertToImage(Syncfusion.Drawing.ImageFormat.Emf);
//Creates a bitmap of specific width and height
Bitmap bitmap = new Bitmap(customWidth, customHeight,
PixelFormat.Format32bppArgb);
//Gets the graphics from image
Graphics graphics = Graphics.FromImage(bitmap);
//Sets the resolution
bitmap.SetResolution(graphics.DpiX, graphics.DpiY);
//Recreates the image in custom size
graphics.DrawImage(System.Drawing.Image.FromStream(stream), new Rectangle(0,
0, bitmap.Width, bitmap.Height));

```

```
//Saves the image as bitmap
bitmap.Save("ImageOutput" + Guid.NewGuid().ToString() + ".jpeg");
//Closes the presentation
pptxDoc.Close();
```

VB.NET

```
'Loads the PowerPoint presentation
Dim pptxDoc As IPresentation = Presentation.Open("Output.pptx")
'Declare variables to hold custom width and height
Dim customWidth As Integer = 1500
Dim customHeight As Integer = 1000
'Converts the slide as image and returns the image stream
Dim stream As Stream =
pptxDoc.Slides(0).ConvertToImage(Syncfusion.Drawing.ImageFormat.Emf)
'Creates a bitmap of specific width and height
Dim bitmap As New Bitmap(customWidth, customHeight,
PixelFormat.Format32bppPArgb)
'Gets the graphics from image
Dim imageGraphics As Graphics = Graphics.FromImage(bitmap)
'Sets the resolution
bitmap.SetResolution(imageGraphics.DpiX, imageGraphics.DpiY)
'Recreates the image in custom size
imageGraphics.DrawImage(System.Drawing.Image.FromStream(stream), New
Rectangle(0, 0, bitmap.Width, bitmap.Height))
'Saves the image as bitmap
bitmap.Save("ImageOutput" + Guid.NewGuid().ToString() + ".jpeg")
'Closes the presentation
pptxDoc.Close()
```

UWP

PowerPoint slides can be converted to images in UWP by using Essential Presentation library. The following assemblies are required in the UWP application to convert the slides as images.

Assembly Name	Short Description
Syncfusion.Presentation.UWP	This assembly contains the core features needed for creating, reading, manipulating a Presentation file.
Syncfusion.OfficeChart.UWP	This assembly contains the Office Chart Object model and core features needed for chart creation.
Syncfusion.OfficeChartToImageConverter.UWP	This assembly is used to convert Office Chart into Image.
Syncfusion.SfChart.UWP	Supporting assembly for Syncfusion.OfficeChartToImageConverter.UWP

The following code example demonstrates how to convert a slide to image in UWP.

C#

```
//Load the presentation file using open picker
FileOpenPicker openPicker = new FileOpenPicker();
```

```
openPicker.FileTypeFilter.Add(".pptx");
StorageFile inputFile = await openPicker.PickSingleFileAsync();
pptxDoc = await Presentation.OpenAsync(inputFile);
//Initialize the 'ChartToImageConverter' instance to convert the charts in
the slides
pptxDoc.ChartToImageConverter = new ChartToImageConverter();
//Pick the folder to save the converted images.
FolderPicker folderPicker = new FolderPicker();
folderPicker.ViewMode = PickerViewMode.Thumbnail;
folderPicker.FileTypeFilter.Add("*");
StorageFolder storageFolder = await folderPicker.PickSingleFolderAsync();
StorageFile imageFile = await storageFolder.CreateFileAsync("Slide1.jpg",
CreationCollisionOption.ReplaceExisting);
//Convert the slide to image.
await slide.SaveAsImageAsync(imageFile);
//Closes the presentation instance
pptxDoc.Close();
```

The following code snippet demonstrates how to convert a PowerPoint slide to image using custom image resolution.

C#

```
//Load the presentation file using open picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.FileTypeFilter.Add(".pptx");
StorageFile inputFile = await openPicker.PickSingleFileAsync();
pptxDoc = await Presentation.OpenAsync(inputFile);
//Initialize the 'ChartToImageConverter' instance to convert the charts in
the slides.
pptxDoc.ChartToImageConverter = new ChartToImageConverter();
//Pick the folder to save the converted images.
FolderPicker folderPicker = new FolderPicker();
folderPicker.ViewMode = PickerViewMode.Thumbnail;
folderPicker.FileTypeFilter.Add("*");
StorageFolder storageFolder = await folderPicker.PickSingleFolderAsync();
StorageFile imageFile = await storageFolder.CreateFileAsync("Slide1.jpg",
CreationCollisionOption.ReplaceExisting);
//Get the stream of the created image file.
StorageFile imageStream = await imageFile.OpenStreamForWriteAsync()
//Creates a new instance for the rendering options to customize the image
resolution.
RenderingOptions renderingOptions = new RenderingOptions();
//Sets the horizontal scaling value for the converted image. The default
value is 1.
renderingOptions.ScaleX = 10F;
//Sets the vertical scaling value for the converted image. The default value
is 1.
renderingOptions.ScaleY = 10F;
//Convert the slide to image with specified resolution.
await slide.SaveAsImageAsync(imageStream, renderingOptions);
//Closes the presentation instance
pptxDoc.Close();
```

The following code snippet demonstrates how to convert a PowerPoint slide to image by passing 'CancellationToken'.

C#

```
//Load the presentation file using open picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.FileTypeFilter.Add(".pptx");
StorageFile inputFile = await openPicker.PickSingleFileAsync();
pptxDoc = await Presentation.OpenAsync(inputFile);
//Initialize the ChartToImageConverter instance to convert the charts in the
slides.
pptxDoc.ChartToImageConverter = new ChartToImageConverter();
//Pick the folder to save the converted images.
FolderPicker folderPicker = new FolderPicker();
folderPicker.ViewMode = PickerViewMode.Thumbnail;
folderPicker.FileTypeFilter.Add("*");
StorageFolder storageFolder = await folderPicker.PickSingleFolderAsync();
//Create a cancellation token to cancel the image rendering instantly.
CancellationTokensource cancellationToken = new CancellationTokensource();
//Convert the slide to image.
int slideNumber = 1;
foreach (ISlide slide in pptxDoc.Slides)
{
    StorageFile imageFile = await storageFolder.CreateFileAsync("Slide" +
slideNumber++ + ".jpg", CreationCollisionOption.ReplaceExisting);
    await slide.SaveAsImageAsync(imageFile, cancellationToken.Token);
}
//Close the Presentation instance
pptxDoc.Close();
```

Note: 1. PowerPoint Presentation to image conversion is supported in Blazor server-side application alone and is not supported in Blazor client-side application.

2. Instance of **ChartToImageConverter** class is mandatory to convert the charts present in the Presentation to image. Otherwise, the charts in the presentation are not exported to the converted image

3. **ChartToImageConverter** is supported from .NET Framework 4.0 onward

4. The assembly "Syncfusion.SfChart.WPF" is non compliance with FIPS(Federal Information Processing Standard) algorithm policy.

5. EMF images in the PowerPoint slides will not be converted in UWP due to platform limitation.

6. Radial gradient, rectangular gradient and path gradient brushes are not supported in UWP due to platform limitation. These brushes are rendered as linear gradient brush in our UWP slide to image conversion.

Font substitution for unavailable fonts

When a font used in a PowerPoint presentation is unavailable in the environment where it is converted to image, then the library substitutes the 'Microsoft Sans Serif' as a default font for text rendering. This leads to a difference in text layouts of PowerPoint presentation and the converted image. To avoid this, the Essential Presentation library allows you to set an alternate font for the missing font used in the PowerPoint presentation.

Note: Font substitution for Unavailable fonts is not supported in UWP platform.

The following code sample demonstrates how to set a substitute font for a missing font while converting a PowerPoint presentation to image.

C#

```
//Load the PowerPoint presentation and convert to image
using (IPresentation pptxDoc = Presentation.Open("Sample.pptx"))
{
    //Initialize 'ChartToImageConverter' to convert charts in the slides, and
    this is optional
    pptxDoc.ChartToImageConverter = new ChartToImageConverter();
    // Initializes the 'SubstituteFont' event to set the replacement font
    pptxDoc.FontSettings.SubstituteFont += FontSettings_SubstituteFont;
    //Converts the first slide into image
    Image image =
    pptxDoc.Slides[0].ConvertToImage(Syncfusion.Drawing.ImageType.Metafile);
    //Saves the image as file
    image.Save("slide1.png");
    //Disposes the image
    image.Dispose();
}
/// <summary>
/// Sets the alternate font when a specified font is unavailable in the
production environment
/// </summary>
/// <param name="sender">FontSettings type of the Presentation in which the
specified font is used but unavailable in production environment. </param>
/// <param name="args">Retrieves the unavailable font name and receives the
substitute font name for conversion. </param>
private static void FontSettings_SubstituteFont(object sender,
SubstituteFontEventArgs args)
{
    if (args.OriginalFontName == "Arial Unicode MS")
        args.AlternateFontName = "Arial";
    else
        args.AlternateFontName = "Times New Roman";
}
```

VB.NET

```
'Load the PowerPoint presentation and convert to image
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Initialize 'ChartToImageConverter' to convert charts in the slides, and
this is optional
pptxDoc.ChartToImageConverter = New ChartToImageConverter()
'Initializes the 'SubstituteFont' event to set the replacement font
AddHandler pptxDoc.FontSettings.SubstituteFont, AddressOf SubstituteFont
'Convert the PowerPoint presentation to image.
Dim image As Image =
pptxDoc.Slides(0).ConvertToImage(Syncfusion.Drawing.ImageType.Metafile)
'Save the image.
image.Save("slide1.png")
'Dispose the Presentation instance
pptxDoc.Dispose()
'Dispose the image
```

```

image.Dispose()
''' <summary>
''' Sets the alternate font when a specified font is unavailable in the
production environment
''' </summary>
''' <param name="sender">FontSettings type of the Presentation in which the
specified font is used but unavailable in production environment. </param>
''' <param name="args">Retrieves the unavailable font name and receives the
substitute font name for conversion. </param>
Private Sub SubstituteFont(ByVal sender As Object, ByVal args As
SubstituteFontEventArgs)
' Sets the alternate font when a specified font is not installed in the
production environment
If args.OriginalFontName = "Arial Unicode MS" Then
args.AlternateFontName = "Arial"
Else
args.AlternateFontName = "Times New Roman"
End If
End Sub

```

Presentation to PDF conversion

PowerPoint allows you to convert an entire Presentation or a single slide into PDF document. Refer to the following links for assemblies/nuget packages required based on platforms to convert PowerPoint document into PDF.

- [Assemblies Information](#)
- [NuGet Information](#)

PresentationToPdfConverter class is responsible for converting an entire Presentation or a slide into PDF. The following code example demonstrates how to convert a PowerPoint presentation to PDF.

C#

```

//Namespaces to perform PPTX to PDF conversion
using Syncfusion.OfficeChartToImageConverter;
using Syncfusion.Presentation;
using Syncfusion.PresentationToPdfConverter;
using Syncfusion.Pdf;
//Opens a PowerPoint Presentation
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Creates an instance of ChartToImageConverter and assigns it to
ChartToImageConverter property of Presentation
pptxDoc.ChartToImageConverter = new ChartToImageConverter();
//Converts the PowerPoint Presentation into PDF document
PdfDocument pdfDocument = PresentationToPdfConverter.Convert(pptxDoc);
//Saves the PDF document
pdfDocument.Save("Sample.pdf");
//Closes the PDF document
pdfDocument.Close(true);
//Closes the Presentation
pptxDoc.Close();

```

VB.NET

```

'Namespaces to perform PPTX to PDF conversion
Imports Syncfusion.OfficeChartToImageConverter
Imports Syncfusion.Presentation
Imports Syncfusion.PresentationToPdfConverter
Imports Syncfusion.Pdf
'Opens a PowerPoint Presentation
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Creates an instance of ChartToImageConverter and assigns it to
ChartToImageConverter property of Presentation
pptxDoc.ChartToImageConverter = New ChartToImageConverter()
'Converts the PowerPoint Presentation into PDF document
Dim pdfDocument As PdfDocument = PresentationToPdfConverter.Convert(pptxDoc)
'Saves the PDF document
pdfDocument.Save("Sample.pdf")
'Closes the PDF document
pdfDocument.Close(True)
'Closes the Presentation
pptxDoc.Close()

```

ASP.NET CORE

```

//Namespaces to perform PPTX to PDF conversion
using Syncfusion.Pdf;
using Syncfusion.Presentation;
using Syncfusion.PresentationToPdfConverter;
using System.IO;
//Load the PowerPoint presentation into stream.
using (FileStream fileStreamInput = new FileStream(@"Template.pptx",
FileMode.Open, FileAccess.Read))
{
//Open the existing PowerPoint presentation with loaded stream.
using (IPresentation pptxDoc = Presentation.Open(fileStreamInput))
{
//Create the MemoryStream to save the converted PDF.
using (MemoryStream pdfStream = new MemoryStream())
{
//Convert the PowerPoint document to PDF document.
using (PdfDocument pdfDocument =
PresentationToPdfConverter.Convert(pptxDoc))
{
//Save the converted PDF document to MemoryStream.
pdfDocument.Save(pdfStream);
pdfStream.Position = 0;
}
}
//Create the output PDF file stream
using (FileStream fileStreamOutput = File.Create("Output.pdf"))
{
//Copy the converted PDF stream into created output PDF stream
pdfStream.CopyTo(fileStreamOutput);
}
}
}
}

```

XAMARIN


```
//Namespaces to perform PPTX to PDF conversion
using Syncfusion.Pdf;
using Syncfusion.Presentation;
using Syncfusion.PresentationToPdfConverter;
using System.IO;
//Load the PowerPoint presentation into stream.
using (FileStream fileStreamInput = new FileStream(@"Template.pptx",
    FileMode.Open, FileAccess.Read))
{
    //Open the existing PowerPoint presentation with loaded stream.
    using (IPresentation pptxDoc = Presentation.Open(fileStreamInput))
    {
        //Create the MemoryStream to save the converted PDF.
        using (MemoryStream pdfStream = new MemoryStream())
        {
            //Convert the PowerPoint document to PDF document.
            using (PdfDocument pdfDocument =
                PresentationToPdfConverter.Convert(pptxDoc))
            {
                //Save the converted PDF document to MemoryStream.
                pdfDocument.Save(pdfStream);
                pdfStream.Position = 0;
            }
            //Create the output PDF file stream
            using (FileStream fileStreamOutput = File.Create("Output.pdf"))
            {
                //Copy the converted PDF stream into created output PDF stream
                pdfStream.CopyTo(fileStreamOutput);
            }
        }
    }
}
```

Note: 1. PowerPoint Presentation to PDF conversion is supported in Blazor server-side application alone and is not supported in Blazor client-side application.

2. Creating an instance of **ChartToImageConverter** class is mandatory to convert the charts present in the Presentation to PDF. Otherwise, the charts are not exported to the converted PDF

3. **ChartToImageConverter** is supported from .NET Framework 4.0 onwards

4. The assembly "Syncfusion.SfChart.WPF" is non compliance with FIPS(Federal Information Processing Standard) algorithm policy.

Customizing the PowerPoint Presentation to PDF conversion

Essential Presentation library provides you the ability to customize the Presentation to PDF conversion with the following options:

- Specify the number of slides per PDF page with 'Handouts' option.
- Convert slides with notes pages to PDF.
- Embed fonts in a PowerPoint file into the converted PDF document to avoid font-related issues across different machines and different platforms.
- Convert a PowerPoint document to PDF with the PDF-A1B conformance standards.
- Specify fallback fonts to be used in place of missing fonts.

- Skip or include hidden slides
- Set the quality of images in the PowerPoint slides to reduce the converted PDF document size.

Font substitution for unavailable fonts

When a font used in a PowerPoint presentation is unavailable in the environment where it is converted to PDF, then the library substitutes the 'Microsoft Sans Serif' as a default font for text rendering. This leads to a difference in text layouts of PowerPoint presentation and the converted PDF document. To avoid this, the Essential Presentation library allows you to set an alternate font for the missing font used in the PowerPoint presentation.

Set alternate font

The following code example demonstrates how to set alternate font name for a missing font while converting a PowerPoint presentation to PDF. The provided alternate font should be installed in the production environment.

C#

```
//Load the PowerPoint presentation and convert to PDF
using (IPresentation pptxDoc = Presentation.Open("Sample.pptx"))
{
    //Initialize 'ChartToImageConverter' to convert charts in the slides, and
    //this is optional
    pptxDoc.ChartToImageConverter = new ChartToImageConverter();
    // Initializes the 'SubstituteFont' event to set the replacement font
    pptxDoc.FontSettings.SubstituteFont += FontSettings_SubstituteFont;
    //Convert the PowerPoint presentation to PDF file
    using (PdfDocument pdfDoc = PresentationToPdfConverter.Convert(pptxDoc))
    {
        //Save the PDF file
        pdfDoc.Save("Sample.pdf");
    }
}

/// <summary>
/// Sets the alternate font when a specified font is unavailable in the
/// production environment
/// </summary>
/// <param name="sender">FontSettings type of the Presentation in which the
/// specified font is used but unavailable in production environment. </param>
/// <param name="args">Retrieves the unavailable font name and receives the
/// substitute font name for conversion. </param>
private static void FontSettings_SubstituteFont(object sender,
SubstituteFontEventArgs args)
{
    if (args.OriginalFontName == "Arial Unicode MS")
        args.AlternateFontName = "Arial";
    else
        args.AlternateFontName = "Times New Roman";
}
```

VB.NET

```
'Load the PowerPoint presentation and convert to PDF
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Initialize 'ChartToImageConverter' to convert charts in the slides, and
this is optional
```

```

pptxDoc.ChartToImageConverter = New ChartToImageConverter()
'Initializes the 'SubstituteFont' event to set the replacement font
AddHandler pptxDoc.FontSettings.SubstituteFont, AddressOf SubstituteFont
'Convert the PowerPoint presentation to PDF file
Dim pdfDoc As PdfDocument = PresentationToPdfConverter.Convert(pptxDoc)
'Save the PDF file.
pdfDoc.Save("Sample.pdf")
'Dispose the PowerPoint presentation instance
pptxDoc.Dispose()
'Dispose the PDF document instance
pdfDoc.Dispose()
''' <summary>
''' Sets the alternate font when a specified font is unavailable in the
production environment
''' </summary>
''' <param name="sender">FontSettings type of the Presentation in which the
specified font is used but unavailable in production environment. </param>
''' <param name="args">Retrieves the unavailable font name and receives the
substitute font name for conversion. </param>
Private Sub SubstituteFont(ByVal sender As Object, ByVal args As
SubstituteFontEventArgs)
' Sets the alternate font when a specified font is not installed in the
production environment
If args.OriginalFontName = "Arial Unicode MS" Then
args.AlternateFontName = "Arial"
Else
args.AlternateFontName = "Times New Roman"
End If
End Sub

```

ASP.NET CORE

```

//Load the PowerPoint presentation as stream
using (FileStream fileStream = new FileStream("Sample.pptx",
FileMode.Create))
{
//Load the PowerPoint presentation from stream
using (IPresentation pptxDoc = Presentation.Open(fileStream))
{
// Initializes the 'SubstituteFont' event to set the replacement font
pptxDoc.FontSettings.SubstituteFont += SubstituteFont;
//Convert the PowerPoint presentation to PDF file
PdfDocument pdfDocument = PresentationToPdfConverter.Convert(pptxDoc);
//Create new instance of file stream
FileStream pdfStream = new FileStream("Output.pdf", FileMode.Create);
//Save the generated PDF to file stream
pdfDocument.Save(pdfStream);
//Release all resources
pdfStream.Dispose();
pdfDocument.Close(true);
}
}
/// <summary>
/// Sets the alternate font when a specified font is unavailable in the
production environment
/// </summary>

```

```

/// <param name="sender">FontSettings type of the Presentation in which the
specified font is used but unavailable in production environment. </param>
/// <param name="args">Retrieves the unavailable font name and receives the
substitute font name for conversion. </param>
private static void SubstituteFont(object sender, SubstituteFontEventArgs
args)
{
if (args.OriginalFontName == "Arial Unicode MS")
args.AlternateFontName = "Arial";
else
args.AlternateFontName = "Times New Roman";
}

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
// Opens the PowerPoint presentation
using (IPresentation pptxDoc =
Presentation.Open(assembly.GetManifestResourceStream("GettingStarted.Assets.
Sample.pptx")))
{
// Initializes the 'SubstituteFont' event to set the replacement font
pptxDoc.FontSettings.SubstituteFont += SubstituteFont;
// Convert PowerPoint presentation to PDF
PdfDocument pdfDocument = PresentationToPdfConverter.Convert(pptxDoc);
// Create new instance of MemoryStream
MemoryStream stream = new MemoryStream();
// Save the Pdf to memory stream
pdfDocument.Save(stream);
stream.Position = 0;
// Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}
/// <summary>
/// Sets the alternate font when a specified font is unavailable in the
production environment
/// </summary>
/// <param name="sender">FontSettings type of the Presentation in which the
specified font is used but unavailable in production environment. </param>
/// <param name="args">Retrieves the unavailable font name and receives the
substitute font name for conversion. </param>
private static void SubstituteFont(object sender, SubstituteFontEventArgs
args)
{
if (args.OriginalFontName == "Arial Unicode MS")
args.AlternateFontName = "Arial";
else
args.AlternateFontName = "Times New Roman";
}

```

Upload font stream

The following code example demonstrates how to upload a font stream for missing font while converting a PowerPoint presentation to PDF. The provided alternate font stream is not mandatory to be installed in the production environment.

C#

```
//Load the PowerPoint presentation and convert to PDF
using (IPresentation pptxDoc = Presentation.Open("Sample.pptx"))
{
    //Initialize 'ChartToImageConverter' to convert charts in the slides, and
    //this is optional
    pptxDoc.ChartToImageConverter = new ChartToImageConverter();
    // Initializes the 'SubstituteFont' event to set the replacement font
    pptxDoc.FontSettings.SubstituteFont += FontSettings_SubstituteFont;
    //Convert the PowerPoint presentation to PDF file
    using (PdfDocument pdfDoc = PresentationToPdfConverter.Convert(pptxDoc))
    {
        //Save the PDF file
        pdfDoc.Save("Sample.pdf");
    }
}

/// <summary>
/// Sets the alternate font stream when a specified font is unavailable in
/// the production environment
/// </summary>
/// <param name="sender">FontSettings type of the Presentation in which the
/// specified font stream is used but unavailable in production environment.
/// </param>
/// <param name="args">Retrieves the unavailable font name and receives the
/// substitute font stream for conversion. </param>
private static void FontSettings_SubstituteFont(object sender,
SubstituteFontEventArgs args)
{
    if (args.OriginalFontName == "Arial" && args.FontStyle == FontStyle.Bold)
        args.AlternateFontStream = new FileStream("cambriab.ttf", FileMode.Open);
    else if (args.OriginalFontName == "Arial" && args.FontStyle ==
        FontStyle.Regular)
        args.AlternateFontStream = new FileStream("BROADW.TTF", FileMode.Open);
    else
        args.AlternateFontStream = new FileStream("COOPBL.TTF", FileMode.Open);
}
```

VB.NET

```
'Load the PowerPoint presentation and convert to PDF
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Initialize 'ChartToImageConverter' to convert charts in the slides, and
this is optional
pptxDoc.ChartToImageConverter = New ChartToImageConverter()
'Initializes the 'SubstituteFont' event to set the replacement font
AddHandler pptxDoc.FontSettings.SubstituteFont, AddressOf SubstituteFont
'Convert the PowerPoint presentation to PDF file
Dim pdfDoc As PdfDocument = PresentationToPdfConverter.Convert(pptxDoc)
'Save the PDF file.
pdfDoc.Save("Sample.pdf")
```

```

'Dispose the PowerPoint presentation instance
pptxDoc.Dispose()
'Dispose the PDF document instance
pdfDoc.Dispose()
''' <summary>
''' Sets the alternate font stream when a specified font is unavailable in
the production environment
''' </summary>
''' <param name="sender">FontSettings type of the Presentation in which the
specified font stream is used but unavailable in production
environment.</param>
''' <param name="args">Retrieves the unavailable font name and receives the
substitute font stream for conversion. </param>
Private Sub SubstituteFont(ByVal sender As Object, ByVal args As
SubstituteFontEventArgs)
' Sets the alternate font when a specified font is not installed in the
production environment
If args.OriginalFontName = "Arial" AndAlso args.FontStyle = FontStyle.Bold
Then
args.AlternateFontStream = New FileStream("cambriab.ttf", FileMode.Open)
args.AlternateFontName = "Arial"
ElseIf args.OriginalFontName = "Arial" AndAlso args.FontStyle =
FontStyle.Regular Then
args.AlternateFontStream = New FileStream("BROADW.TTF", FileMode.Open)
Else
args.AlternateFontStream = New FileStream("COOPBL.TTF", FileMode.Open)
End If
End Sub

```

ASP.NET CORE

```

//Load the PowerPoint presentation as stream
using (FileStream fileStream = new FileStream("Sample.pptx",
FileMode.Create))
{
//Load the PowerPoint presentation from stream
using (IPresentation pptxDoc = Presentation.Open(fileStream))
{
// Initializes the 'SubstituteFont' event to set the replacement font
pptxDoc.FontSettings.SubstituteFont += SubstituteFont;
//Convert the PowerPoint presentation to PDF file
PdfDocument pdfDocument = PresentationToPdfConverter.Convert(pptxDoc);
//Create new instance of file stream
FileStream pdfStream = new FileStream("Output.pdf", FileMode.Create);
//Save the generated PDF to file stream
pdfDocument.Save(pdfStream);
//Release all resources
pdfStream.Dispose();
pdfDocument.Close(true);
}
}
////// <summary>
////// Sets the alternate font stream when a specified font is unavailable in
the production environment
////// </summary>

```

```

/// <param name="sender">FontSettings type of the Presentation in which the
specified font stream is used but unavailable in production environment.
</param>
/// <param name="args">Retrieves the unavailable font name and receives the
substitute font stream for conversion. </param>
private static void FontSettings_SubstituteFont(object sender,
SubstituteFontEventArgs args)
{
if (args.OriginalFontName == "Arial" && args.FontStyle == FontStyle.Bold)
args.AlternateFontStream = new FileStream("cambriab.ttf", FileMode.Open);
else if (args.OriginalFontName == "Arial" && args.FontStyle ==
FontStyle.Regular)
args.AlternateFontStream = new FileStream("BROADW.TTF", FileMode.Open);
else
args.AlternateFontStream = new FileStream("COOPBL.TTF", FileMode.Open);
}

```

XAMARIN

```

// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
// Opens the PowerPoint presentation
using (IPresentation pptxDoc =
Presentation.Open(assembly.GetManifestResourceStream("GettingStarted.Assets.
Sample.pptx")))
{
// Initializes the 'SubstituteFont' event to set the replacement font
pptxDoc.FontSettings.SubstituteFont += SubstituteFont;
// Convert PowerPoint presentation to PDF
PdfDocument pdfDocument = PresentationToPdfConverter.Convert(pptxDoc);
// Create new instance of MemoryStream
MemoryStream stream = new MemoryStream();
// Save the Pdf to memory stream
pdfDocument.Save(stream);
stream.Position = 0;
// Save the stream as a file in the device and invoke it for viewing
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}
/// <summary>
/// Sets the alternate font stream when a specified font is unavailable in
the production environment
/// </summary>
/// <param name="sender">FontSettings type of the Presentation in which the
specified font stream is used but unavailable in production environment.
</param>
/// <param name="args">Retrieves the unavailable font name and receives the
substitute font stream for conversion. </param>
private static void FontSettings_SubstituteFont(object sender,
SubstituteFontEventArgs args)
{
// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
if (args.OriginalFontName == "Arial" && args.FontStyle == FontStyle.Bold)
args.AlternateFontStream =
assembly.GetManifestResourceStream("GettingStarted.Assets.cambriab.ttf");
}

```

```

else if (args.OriginalFontName == "Arial" && args.FontStyle ==
FontStyle.Regular)
args.AlternateFontStream =
assembly.GetManifestResourceStream("GettingStarted.Assets.BROADW.TTF");
else
args.AlternateFontStream =
assembly.GetManifestResourceStream("GettingStarted.Assets.COOPBL.TTF");
}

```

Show Warning for unsupported elements

The Presentation library shows warning message about the unsupported elements such as Metafile images and charts (supported from .NET Standard 2.0) present in the input PowerPoint presentation, during PowerPoint to PDF conversion. It also allows you to cancel or continue the PowerPoint to PDF conversion, when any unsupported elements is present in the input PowerPoint presentation.

The following code example demonstrates how to cancel or continue the PowerPoint presentation to PDF conversion, when an unsupported elements (Metafile and Chart) are present in the input PowerPoint presentation.

C#

```

//Essential Presentation library supports Show warning for unsupported
elements feature in ASP.NET Core, Blazor server-side application and Xamarin
platforms alone.

```

VB.NET

```

//Essential Presentation supports library Show warning for unsupported
elements feature in ASP.NET Core, Blazor server-side application and Xamarin
platforms alone.

```

ASP.NET CORE

```

// Open the file as Stream
using (FileStream pptStream = new FileStream("Template.pptx", FileMode.Open,
FileAccess.Read))
{
//Open the existing PowerPoint presentation with loaded stream.
using (IPresentation pptxDoc = Presentation.Open(pptStream))
{
//Instantiation of PresentationToPdfConverterSettings
PresentationToPdfConverterSettings settings = new
PresentationToPdfConverterSettings();
//Gets all the warnings into the collection
settings.Warning = new DocumentWarning();
//Converts the PowerPoint Presentation into PDF document
using (PdfDocument pdfDocument = PresentationToPdfConverter.Convert(pptxDoc,
settings))
{
//If the PowerPoint to Pdf conversion has been stopped, IsCanceled value
will be True, otherwise false
if (!PresentationToPdfConverter.IsCanceled)
{
//Saves the PDF file

```



```

using (FileStream outputStream = new FileStream("Output.pdf",
    FileMode.OpenOrCreate, FileAccess.ReadWrite))
{
    pdfDocument.Save(outputStream);
    outputStream.Position = 0;
}
}
else
{
    Console.WriteLine("PowerPoint to PDF conversion is stopped , please press
any key to exit the application");
    Console.ReadKey();
}
}
}
}
/// <summary>
/// DocumentWarning class implements the IWarning interface
/// </summary>
/// <seealso cref="IWarning" />
public class DocumentWarning : IWarning
{
    /// <summary>
    /// Gets the Boolean value whether to continue conversion or not
    /// </summary>
    /// <param name="warningInfo">Collection of warnings</param>
    /// <returns></returns>
    public bool ShowWarnings(List<WarningInfo> warningInfo)
    {
        //By default to perform the PowerPoint to PDF conversion by setting the
        isContinueConversion as true.
        bool isContinueConversion = true;
        foreach (WarningInfo warning in warningInfo)
        {
            //Since there are warnings in the PowerPoint presentation the value of
            isContinueConversion will be set as false.
            isContinueConversion = false;
            //Print the description of the Warning
            Console.WriteLine(warning.Description);
            if (warning.Description.Contains("Metafile") ||
            warning.Description.Contains("Chart"))
            {
                Console.WriteLine("Type [Y] if you want Do you want to continue Presentation
to Pdf conversion or Type [N] to cancel the conversion");
                String confrimation = Console.ReadLine();
                //Based on warning.WarningType enumeration, you can do your manipulation.
                //Skips the PowerPoint to Pdf conversion by setting isContinueConversion
                value as false.
                //Continue the PowerPoint to PDF conversion by setting the
                isContinueConversion as true.
                if (confrimation.ToLower().Equals("y"))
                isContinueConversion = true;
                else
                isContinueConversion = false;
            }
        }
        return isContinueConversion;
    }
}

```

```
}
}
```

XAMARIN

```
// "App" is the class of Portable project.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
// Open the file as Stream
using (FileStream pptStream = new FileStream("Template.pptx", FileMode.Open,
FileAccess.Read))
{
    // Open the existing PowerPoint presentation with loaded stream.
    using (IPresentation pptxDoc = Presentation.Open(pptStream))
    {
        // Instantiation of PresentationToPdfConverterSettings
        PresentationToPdfConverterSettings settings = new
        PresentationToPdfConverterSettings();
        // Gets all the warnings into the collection
        settings.Warning = new DocumentWarning();
        // Converts the PowerPoint Presentation into PDF document
        using (PdfDocument pdfDocument = PresentationToPdfConverter.Convert(pptxDoc,
        settings))
        {
            // If the PowerPoint to Pdf conversion has been stopped, IsCanceled value
            // will be True and PDF document will not get generated, Otherwise false, then
            // the PDF document will get generated and can be viewed.
            if (!PresentationToPdfConverter.IsCanceled)
            {
                // Saves the PDF file
                using (FileStream outputStream = new FileStream("Output.pdf",
                FileMode.OpenOrCreate, FileAccess.ReadWrite))
                {
                    pdfDocument.Save(outputStream);
                    outputStream.Position = 0;
                    // Save the stream as a file in the device and invoke it for viewing
                    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
                    "application/pdf", outputStream);
                }
            }
        }
    }
}

/// <summary>
/// DocumentWarning class implements the IWarning interface
/// </summary>
/// <seealso cref="IWarning" />
public class DocumentWarning : IWarning
{
    /// <summary>
    /// Gets the Boolean value whether to continue conversion or not
    /// </summary>
    /// <param name="warningInfo">Collection of warnings</param>
    /// <returns></returns>
    public bool ShowWarnings(List<WarningInfo> warningInfo)
    {

```

```

//By default to perform the PowerPoint to PDF conversion by setting the
isContinueConversion as true.
bool isContinueConversion = true;
foreach (WarningInfo warning in warningInfo)
{
    //Since there are warnings in the PowerPoint presentation the value of
isContinueConversion will be set as false.
    isContinueConversion = false;
    //Based on warning.WarningType enumeration, you can do your manipulation.
    if (warning.Description.Contains("Metafile") ||
        warning.Description.Contains("Chart"))
    {
        //Continue the PowerPoint to PDF conversion by setting the
isContinueConversion as true.
        isContinueConversion = true;
    }
}
return isContinueConversion;
}
}

```

Handouts

The Presentation library allows you to convert a PowerPoint presentation to PDF document with 'Handouts' option. Thus, the library allows selecting the number of slides to be included per PDF page. This helps converting multiple PowerPoint slides within a single PDF page.

The following code sample demonstrates how to convert a PowerPoint presentation to PDF document with 'Handouts' property.

C#

```

//Load the PowerPoint presentation to convert.
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Enable the handouts and number of pages per slide options in converter
settings.
PresentationToPdfConverterSettings pdfConverterSettings = new
PresentationToPdfConverterSettings();
pdfConverterSettings.PublishOptions = PublishOptions.Handouts;
pdfConverterSettings.SlidesPerPage = SlidesPerPage.Nine;
//Convert the documents by passing the PDF conversion settings as parameter.
PdfDocument pdfDoc = PresentationToPdfConverter.Convert(pptxDoc,
pdfConverterSettings);
//Save the converted PDF document.
pdfDoc.Save("Sample.pdf");
//Close the presentation instance
pptxDoc.Close();
//Close the PDF instance
pdfDoc.Close();

```

VB.NET

```

'Load the PowerPoint presentation to convert.
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Enable the handouts and number of pages per slide options in converter
settings.

```

```

Dim pdfConverterSettings As PresentationToPdfConverterSettings = new
PresentationToPdfConverterSettings()
pdfConverterSettings.PublishOptions = PublishOptions.Handouts
pdfConverterSettings.SlidesPerPage = SlidesPerPage.Nine
'Convert the documents by passing the PDF conversion settings as parameter.
Dim pdfDoc As PdfDocument = PresentationToPdfConverter.Convert(pptxDoc,
pdfConverterSettings)
'Save the converted PDF document.
pdfDoc.Save("Sample.pdf")
'Close the presentation instance
pptxDoc.Close()
'Close the PDF instance
pdfDoc.Close()

```

Notes pages

The Presentation library allows you to convert a PowerPoint presentation to PDF document with 'notes pages' option, which will includes the notes content while PDF conversion.

The following code sample demonstrates how to convert a PowerPoint presentation to PDF with 'notes pages'.

C#

```

//Load the PowerPoint presentation to convert.
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Enable the include hidden slides option in converter settings.
PresentationToPdfConverterSettings pdfConverterSettings = new
PresentationToPdfConverterSettings();
pdfConverterSettings.PublishOptions = PublishOptions.NotesPages;
//Convert the documents by passing the settings as parameter.
PdfDocument pdfDoc = PresentationToPdfConverter.Convert(pptxDoc,
pdfConverterSettings);
//Save the converted PDF file.
pdfDoc.Save("Sample.pdf");
//Close the presentation instance
pptxDoc.Close();
//Close the PDF instance
pdfDoc.Close();

```

VB.NET

```

'Load the PowerPoint presentation to convert.
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Enable the include hidden slides option in converter settings.
Dim pdfConverterSettings As PresentationToPdfConverterSettings
pdfConverterSettings = new PresentationToPdfConverterSettings()
pdfConverterSettings.PublishOptions = PublishOptions.NotesPages
'Convert the documents by passing the settings as parameter.
Dim pdfDoc As PdfDocument = PresentationToPdfConverter.Convert(pptxDoc,
pdfConverterSettings)
'Save the converted PDF file.
pdfDoc.Save("Sample.pdf")
'Close the presentation instance
pptxDoc.Close()
'Close the PDF instance

```

```
pdfDoc.Close()
```

Include hidden slides

The PowerPoint presentation supports hiding the slides in a presentation document. The hidden slides will not be included in the converted PDF document, by default. The Presentation library provides support to include or exclude the hidden slides while converting a PowerPoint presentation to PDF document.

The following code sample demonstrates how to include the hidden slides while converting a PowerPoint presentation to PDF document.

C#

```
//Load the PowerPoint presentation to convert.
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Enable or disable including the hidden slides option in converter settings.
PresentationToPdfConverterSettings pdfConverterSettings = new
PresentationToPdfConverterSettings();
pdfConverterSettings.ShowHiddenSlides = true;
//Convert the documents by passing the settings as parameter.
PdfDocument pdfDoc = PresentationToPdfConverter.Convert(pptxDoc,
pdfConverterSettings);
//Save the converted PDF file.
pdfDoc.Save("Sample.pdf");
//Close the presentation instance
pptxDoc.Close();
//Close the PDF instance
pdfDoc.Close();
```

VB.NET

```
'Load the PowerPoint presentation to convert.
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Enable or disable including the hidden slides option in converter settings.
Dim pdfConverterSettings As PresentationToPdfConverterSettings = new
PresentationToPdfConverterSettings()
pdfConverterSettings.ShowHiddenSlides = true
'Convert the documents by passing the settings as parameter.
Dim pdfDoc As PdfDocument = PresentationToPdfConverter.Convert(pptxDoc,
pdfConverterSettings)
'Save the converted PDF file.
pdfDoc.Save("Sample.pdf")
'Close the presentation instance
pptxDoc.Close()
'Close the PDF instance
pdfDoc.Close()
```

PDF Conformance

Essential Presentation currently supports following PDF conformances while converting a PowerPoint document to PDF.

- PDF/A-1b conformance

- PDF/X-1a conformance

Note: 1. To know more details about PDF/A standard refer

<https://en.wikipedia.org/wiki/PDF/A#Description>

2. To know more details about PDF/X standard refer <https://en.wikipedia.org/wiki/PDF/X>

The following code sample demonstrates how to set the PDF conformance level while PowerPoint presentation to PDF conversion.

C#

```
//Load the PowerPoint document
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Initialize the conversion settings
PresentationToPdfConverterSettings pdfConverterSettings = new
PresentationToPdfConverterSettings();
//Set the Pdf conformance level to A1B
pdfConverterSettings.PdfConformanceLevel = PdfConformanceLevel.Pdf_A1B;
//Convert the PowerPoint document to PDF
PdfDocument pdfDoc =
PresentationToPdfConverter.Convert(pptxDoc,pdfConverterSettings);
//Save the converted PDF file.
pdfDoc.Save("Sample.pdf");
//Close the presentation instance
pptxDoc.Close();
//Close the PDF instance
pdfDoc.Close();
```

VB.NET

```
'Load the PowerPoint presentation to convert.
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Enable or disable including the hidden slides option in converter settings.
Dim pdfConverterSettings As PresentationToPdfConverterSettings = new
PresentationToPdfConverterSettings()
'Set the Pdf conformance level to A1B
pdfConverterSettings.PdfConformanceLevel = PdfConformanceLevel.Pdf_A1B
'Convert the documents by passing the settings as parameter.
Dim pdfDoc As PdfDocument = PresentationToPdfConverter.Convert(pptxDoc,
pdfConverterSettings)
'Save the converted PDF file.
pdfDoc.Save("Sample.pdf")
'Close the presentation instance
pptxDoc.Close()
'Close the PDF instance
pdfDoc.Close()
```

Chart quality

The Presentation library provides an option to decide the quality of the charts to optimize the converted PDF document size.

Note: The default 'ScalingMode' for charts is 'ScalingMode.Normal'.

Setting the 'Best' scaling mode will improve the quality of the converted charts and increase the converted PDF document size.

The following code sample demonstrates how to set the quality of the charts while PowerPoint presentation to PDF conversion

C#

```
//Load the PowerPoint presentation to convert.
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Create an instance of ChartToImageConverter and assigns it to
ChartToImageConverter property of Presentation
pptxDoc.ChartToImageConverter = new ChartToImageConverter();
//Sets the scaling mode of the chart to best.
pptxDoc.ChartToImageConverter.ScalingMode = ScalingMode.Best;
//Convert the documents by passing the PDF conversion settings as parameter.
PdfDocument pdfDoc = PresentationToPdfConverter.Convert(pptxDoc);
//Save the converted PDF document.
pdfDoc.Save("Sample.pdf");
//Close the presentation instance
pptxDoc.Close();
//Close the PDF instance
pdfDoc.Close();
```

VB.NET

```
'Load the PowerPoint presentation to convert.
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Create an instance of ChartToImageConverter and assigns it to
ChartToImageConverter property of Presentation
pptxDoc.ChartToImageConverter = new ChartToImageConverter()
'Sets the scaling mode of the chart to best.
pptxDoc.ChartToImageConverter.ScalingMode = ScalingMode.Best
'Convert the documents by passing the PDF conversion settings as parameter.
Dim pdfDoc As PdfDocument = PresentationToPdfConverter.Convert(pptxDoc)
'Save the converted PDF document.
pdfDoc.Save("Sample.pdf")
'Close the presentation instance
pptxDoc.Close()
'Close the PDF instance
pdfDoc.Close()
```

Optimizing the converted PDF document size

Optimizing the identical images

The presentation library allows you to optimize the memory usage for the duplicate images while converting the PowerPoint presentation to PDF document to reduce the document size.

The following code sample demonstrates how to optimize the duplicate images while converting a PowerPoint presentation to PDF document.

C#

```
//Load the PowerPoint presentation to convert.
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Enable the include hidden slides option in converter settings.
```

```

PresentationToPdfConverterSettings pdfConverterSettings = new
PresentationToPdfConverterSettings();
//Set the flag to optimize the identical images
pdfConverterSettings.OptimizeIdenticalImages = true;
//Convert the documents by passing the settings as parameter.
PdfDocument pdfDoc = PresentationToPdfConverter.Convert(pptxDoc,
pdfConverterSettings);
//Save the converted PDF file.
pdfDoc.Save("Sample.pdf");
//Close the presentation instance
pptxDoc.Close();
//Close the PDF instance
pdfDoc.Close();

```

VB.NET

```

'Load the PowerPoint presentation to convert.
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Enable the include hidden slides option in converter settings.
Dim pdfConverterSettings As PresentationToPdfConverterSettings = new
PresentationToPdfConverterSettings()
'Set the flag to optimize the identical images
pdfConverterSettings.OptimizeIdenticalImages = true
'Convert the documents by passing the settings as parameter.
Dim pdfDoc As PdfDocument = PresentationToPdfConverter.Convert(pptxDoc,
pdfConverterSettings)
Save the converted PDF file.
pdfDoc.Save("Sample.pdf")
Close the presentation instance
pptxDoc.Close()
Close the PDF instance
pdfDoc.Close()

```

Optimizing the image quality and resolution

You may have high resolution images in the PowerPoint slides which will impact the size of the converted PDF document. The Presentation library allows you to optimize the document size, by adjusting the quality and resolution of the images while converting the PowerPoint presentation to PDF document.

The following code sample demonstrates how to optimize the image quality and resolution while converting a PowerPoint presentation to PDF document.

C#

```

//Load the PowerPoint presentation to convert.
IPresentation pptxDoc = Presentation.Open("Sample.pptx");
//Enable the include hidden slides option in converter settings.
PresentationToPdfConverterSettings pdfConverterSettings = new
PresentationToPdfConverterSettings();
//Set the image resolution
pdfConverterSettings.ImageResolution = 100;
//Set the image quality
pdfConverterSettings.ImageQuality = 100;
//Convert the documents by passing the settings as parameter.

```



```
PdfDocument pdfDoc = PresentationToPdfConverter.Convert(pptxDoc,
pdfConverterSettings);
//Save the converted PDF file.
pdfDoc.Save("Sample.pdf");
//Close the presentation instance
pptxDoc.Close();
//Close the PDF instance
pdfDoc.Close();
```

VB.NET

```
'Load the PowerPoint presentation to convert.
Dim pptxDoc As IPresentation = Presentation.Open("Sample.pptx")
'Enable the include hidden slides option in converter settings.
Dim pdfConverterSettings As PresentationToPdfConverterSettings = new
PresentationToPdfConverterSettings()
'Set the image resolution
pdfConverterSettings.ImageResolution = 100
'Set the image quality
pdfConverterSettings.ImageQuality = 100
'Convert the documents by passing the settings as parameter.
Dim pdfDoc As PdfDocument = PresentationToPdfConverter.Convert(pptxDoc,
pdfConverterSettings)
'Save the converted PDF file.
pdfDoc.Save("Sample.pdf")
'Close the presentation instance
pptxDoc.Close()
'Close the PDF instance
pdfDoc.Close()
```

PowerPoint to PDF conversion in Azure platform

The Syncfusion PowerPoint library supports converting the PowerPoint document to PDF in Azure platform. The following code sample demonstrates how to convert a PowerPoint presentation to PDF in Azure platform.

C#

```
//Load the PowerPoint document
IPresentation pptxDoc = Presentation.Open("Table.pptx");
//Initialize the conversion settings
PresentationToPdfConverterSettings pdfConverterSettings = new
PresentationToPdfConverterSettings();
//Enable the portable rendering.
pdfConverterSettings.EnablePortableRendering = true;
//Convert the PowerPoint document to PDF
PdfDocument pdfDoc =
PresentationToPdfConverter.Convert(pptxDoc, pdfConverterSettings);
//Save the converted PDF file.
pdfDoc.Save("Sample.pdf");
//Close the presentation instance
pptxDoc.Close();
//Close the PDF instance
pdfDoc.Close();
```

VB.NET

```

'Load the PowerPoint document
Dim pptxDoc As IPresentation = Presentation.Open("Table.pptx")
'Initialize the conversion settings
Dim pdfConverterSettings As PresentationToPdfConverterSettings = new
PresentationToPdfConverterSettings()
'Enable the portable rendering.
pdfConverterSettings.EnablePortableRendering = true
'Convert the PowerPoint document to PDF
Dim pdfDoc As PdfDocument =
PresentationToPdfConverter.Convert(pptxDoc,pdfConverterSettings)
'Save the converted PDF file.
pdfDoc.Save("Sample.pdf")
'Close the presentation instance
pptxDoc.Close()
'Close the PDF instance
pdfDoc.Close()

```

Supported and Unsupported Features

Document Content Features	Windows Forms	WPF	ASP.NET	ASP.Net MVC	Xamarin. Forms	UWP	Blazor	
							Client side application	Server side and hosted application
Create PowerPoint Presentation from Scratch or editing an existing Presentation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Open existing PowerPoint Presentation from the file system or stream	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Save PowerPoint presentation to a local file, stream or stream it to the client browser	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Create, access and modify paragraphs and text	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Create, access and modify bullets and numbering	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Create, access and modify images	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Create, access and modify tables, rows and cells	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Create, access and modify shapes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Create, access and modify charts	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Access and modify Placeholders	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Create, access and modify SmartArt diagrams	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Copy and move Presentation elements between Presentations	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Merge multiple	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

PowerPoint Presentations								
Encrypt and open password protected Presentations	Yes	Yes	Yes	Yes	No	Yes	No	No
Iterate over slide content	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Access and modify built-in document properties	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Create, access and modify custom document properties	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Convert PowerPoint Presentations into PDF	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Convert PowerPoint slides into standard image formats	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Create, access, modify and remove Notes Slide	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

FAQ's

1. Why I get an exception when trying to load a PPT file?

The current version of Presentation library supports only .PPTX format - Microsoft Office 2007 and later version.

2. Is it possible to print the Presentation slides?

Yes, you can print the PowerPoint presentations by using its ability to convert the slides as images and by using the [PrintDocument](#) class. For more details, refer to [Printing](#)

3. Does adding audio and video to a Presentation is supported?

At present, there is no support to add audio and video to Presentation by using Essential Presentation library.

4. What measure does Essential Presentation use to add slide elements such as textbox, shape, picture and charts?

We use Points to add any slide elements in a Presentation.

5. Does Essential Presentation supports cloning a slide in the Presentation?

Yes, Essential Presentation library supports cloning as follows:

- Slide in the Presentation can be cloned from one Presentation to another or within a same Presentation.
 - An entire Presentation can also be cloned as an independent copy of the original.
- 6. Could not find Syncfusion.OfficeChartToImageConverter assembly in .NET 3.5 Framework, does it mean there is no support for chart conversion in this framework?**

Yes, OfficeChartToImageConverter assembly is not supported in .NET 3.5 Framework and it is available from .NET 4.0 Framework.

7. Can chart data be refreshed?

Yes, Essential Presentation supports refreshing the chart data. For more details, refer to [Working with charts](#)

8. Is it possible to convert 3D charts to PDF or image?

Current version of the Essential Presentation library does not provide support for converting 3D charts to PDF or image format.

9. How to improve the image quality while converting the Presentation slides to image?

You can improve the quality of converted images by specifying the image resolution. Refer – [Converting PowerPoint presentation to Images](#)

XlsIO

Overview

Essential XlsIO is a native **.NET** class library that can be used to create and modify **Microsoft Excel** files by using C#, VB.NET and managed C++ code. It is a non-UI component that provides a full-fledged object model that facilitates accessing & manipulating the spreadsheets without any dependency of Microsoft Office COM libraries & Microsoft Office.

The library can be used in Windows Forms, WPF, UWP, ASP.NET Web Forms, ASP.NET MVC, ASP.NET Core, Xamarin and Blazor applications.

Key Features

- Support to [create Excel documents](#) from scratch.
- Support to [modify](#) existing Excel documents.
- Support to [import and export](#) data.
- Supports Excel [formulas](#).
- Ability to create and manipulate [Conditional Formats](#) and [Data Validations](#).
- Create and manipulate [Charts](#) and [Sparklines](#).
- Supports creation and manipulation of [Tables](#), [Pivot Tables](#) and [Pivot Charts](#).
- Fill data using [Template Markers](#).
- [Clone and merge](#) Excel worksheets.
- [Encrypt and decrypt](#) Excel documents.
- Convert Excel [worksheets to images](#).
- Convert [chart in a worksheet to image](#).
- Supports [data filtering](#) and [data sorting](#).
- Support to insert [Images](#) and [AutoShapes](#).
- Support to add [Comments](#) and [Form Controls](#).
- Support to add [Ole objects](#) and [Group Shapes](#).
- Ability to convert Excel [workbook to PDF](#).
- Converts Excel [worksheet to HTML](#).
- Support to open and create [CSV](#) files.
- Support to open and create SpreadsheetML files.
- Access the [Custom document properties](#) of Excel file.
- Access the Built-in document properties.

Compatible Microsoft Excel Versions

- Microsoft Excel 97-2003
- Microsoft Excel 2007
- Microsoft Excel 2010
- Microsoft Excel 2013
- Microsoft Excel 2016
- Microsoft Excel 2019

Assemblies Required

The following assemblies need to be referenced in your application based on the platform.

Platform(s)	Assembly
-------------	----------

WPF, Windows Forms, ASP. NET and ASP.NET MVC	Syncfusion.XlsIO.Base Syncfusion.Compression.Base
Windows Forms and WPF (Client Profile)	Syncfusion.XlsIO.ClientProfile Syncfusion.Compression.Base
Universal Windows Platform	Syncfusion.XlsIO.UWP
.NET Core, Xamarin and Blazor	Syncfusion.XlsIO.Portable Syncfusion.Compression.Portable
WinRT (Windows Store applications)	Syncfusion.XlsIO.WinRT
Windows Phone 8	Syncfusion.XlsIO.WP8 Syncfusion.Compression.WP8
Windows Phone 8.1 Silverlight	Syncfusion.XlsIO.WPSilverlight Syncfusion.Compression.WPSilverlight
Windows Phone 8.1 WinRT	Syncfusion.XlsIO.WP Syncfusion.Compression.WP
Silverlight	Syncfusion.XlsIO.Silverlight Syncfusion.Compression.Silverlight
ASP.NET (Classic)	Syncfusion.XlsIO.Web Syncfusion.Compression.Base
ASP.NET MVC (Classic)	Syncfusion.XlsIO.MVC Syncfusion.Compression.Base
Universal (Classic)	Syncfusion.XlsIO.Universal

Note: Starting with v16.2.0.x, if you reference Syncfusion assemblies from trial setup or from the NuGet feed, you also have to add "Syncfusion.Licensing" assembly reference and include a license key in your projects. Please refer to this [link](#) to know about registering Syncfusion license key in your applications to use our components.

Note: Syncfusion components are available in nuget.org

Converting Excel document to PDF

For converting an Excel document to PDF, the following assemblies need to be referenced in your application.

Platform(s)	Assembly
WPF, Windows Forms, ASP. NET and ASP.NET MVC	Syncfusion.XlsIO.Base Syncfusion.Compression.Base Syncfusion.Pdf.Base Syncfusion.ExcelToPDFConverter.Base
Windows Forms and WPF (Client Profile)	Syncfusion.XlsIO.ClientProfile Syncfusion.Compression.Base

	Syncfusion.Pdf.ClientProfile Syncfusion.ExcelToPDFConverter.ClientProfile
UWP, .NET Core, Xamarin and Blazor (Server-Side)	Syncfusion.Compression.Portable Syncfusion.XlsIO.Portable Syncfusion.Pdf.Portable Syncfusion.SkiaSharpHelper.Portable Syncfusion.XlsIORenderer.Portable

Note: Excel to PDF conversion is supported from .NET Framework 2.0 and .NET Standard 1.4 onwards.

Converting Excel Worksheet to Image

For converting an Excel worksheet to image, the following assemblies need to be referenced in your application.

Platform(s)	Assembly
Windows Forms, WPF, ASP.NET and ASP.NET MVC	Syncfusion.Compression.Base Syncfusion.XlsIO.Base
UWP, .NET Core, Xamarin and Blazor (Server-Side)	Syncfusion.Compression.Portable Syncfusion.XlsIO.Portable Syncfusion.SkiaSharpHelper.Portable Syncfusion.XlsIORenderer.Portable

Note: Worksheet to image conversion is supported from .NET Framework 2.0 and .NET Standard 1.4 onwards.

Converting Excel Chart to Image

For converting an Excel chart to image, the following assemblies need to be referenced in your application.

Platform(s)	Assembly
WPF, Windows Forms, ASP. NET and ASP.NET MVC	Syncfusion.Compression.Base Syncfusion.XlsIO.Base Syncfusion.ExcelChartToImageConverter.WPF Syncfusion.SfChart.WPF
UWP, .NET Core, Xamarin and Blazor (Server-Side)	Syncfusion.Compression.Portable Syncfusion.XlsIO.Portable Syncfusion.SkiaSharpHelper.Portable Syncfusion.XlsIORenderer.Portable

Note: Chart to image conversion is supported from .NET Framework 4.0 and .NET Standard 2.0 onwards.

Note: Starting with v16.2.0.x, if you reference Syncfusion assemblies from trial setup or from the NuGet feed, you also have to include a license key in your projects. Please refer to this [link](#) to know about registering Syncfusion license key in your applications to use our components.

NuGet Packages Required

To work with Excel documents, the following NuGet packages need to be installed in your application.

Platform(s)	NuGet Package
Windows Forms, Console Application (Targeting .NET Framework)	Syncfusion.XlsIO.WinForms.nupkg
WPF	Syncfusion.XlsIO.Wpf.nupkg
.NET Framework 3.5 or 4.0 Client Profile	Syncfusion.XlsIO.ClientProfile.nupkg
ASP.NET Web Forms, ASP.NET Core (Targeting .NET Framework)	Syncfusion.XlsIO.AspNet.nupkg
ASP.NET MVC4	Syncfusion.XlsIO.AspNet.Mvc4.nupkg
ASP.NET MVC5	Syncfusion.XlsIO.AspNet.Mvc5.nupkg
UWP	Syncfusion.XlsIO.UWP.nupkg
ASP.NET Core (Targeting .NET Core), Console Application	[Syncfusion.XlsIO.Net.Core.nupkg](https://www.nuget.org/packages/Syncfusion.XlsIO.Net.Core/)

(Targeting .NET Core)	
Xamarin	Syncfusion.Xamarin.XlsIO.nupkg
Blazor	{{'Syncfusion.XlsIO.Net.Core.nupkg' Â markdownify }}

Note: Starting with v16.2.0.x, if you reference Syncfusion assemblies from trial setup or from the NuGet feed, you also have to add "Syncfusion.Licensing" assembly reference and include a license key in your projects. Please refer to this [link](#) to know about registering Syncfusion license key in your applications to use our components.

Note: From the Essential Studio 2018 Volume 3 release(v16.3.0.21), Syncfusion has changed some of the NuGet package names to search and find the required Syncfusion NuGet packages in nuget.org easily based on the control and its platforms.

Converting Excel document into PDF

For converting Excel document into PDF, the following NuGet packages need to be installed in your application.

Platform(s)	NuGet Package
Windows Forms, Console Application (Targeting .NET Framework)	Syncfusion.ExcelToPDFConverter.WinForms.nupkg
WPF	Syncfusion.ExcelToPDFConverter.Wpf.nupkg
.NET Framework 3.5 or 4.0 Client Profile	Syncfusion.ExcelToPdfConverter.ClientProfile.nupkg
ASP.NET Web Forms, ASP.NET Core (Targeting .NET Framework)	Syncfusion.ExcelToPDFConverter.AspNet.nupkg

ASP.NET MVC4	Syncfusion.ExcelToPDFConverter.AspNet.Mvc4.nupkg
ASP.NET MVC5	Syncfusion.ExcelToPDFConverter.AspNet.Mvc5.nupkg
UWP, ASP.NET Core (Targeting .NET Core), Console Application (Targeting .NET Core)	[Syncfusion.XlsIORenderer.Net.Core.nupkg](https://www.nuget.org/packages/Syncfusion.XlsIORenderer.Net.Core/)
Xamarin	Syncfusion.Xamarin.XlsIORenderer.nupkg
Blazor (Server-Side)	{{' Syncfusion.XlsIORenderer.Net.Core.nupkg ' markdownify }}

Note: Excel to PDF conversion is supported from .NET Framework 2.0 and .NET Standard 1.4 onwards.

Converting Excel Worksheet to Image

For converting an Excel worksheet to image, the following NuGet packages need to be installed in your application.

Platform(s)	NuGet Package
Windows Forms, Console Application (Targeting .NET Framework)	Syncfusion.XlsIO.WinForms.nupkg
WPF	Syncfusion.XlsIO.Wpf.nupkg
.NET Framework 3.5 or	Syncfusion.XlsIO.ClientProfile.nupkg

4.0 Client Profile	
ASP.NET Web Forms, ASP.NET Core (Targeting .NET Framework)	Syncfusion.XlsIO.AspNet.nupkg
ASP.NET MVC4	Syncfusion.XlsIO.AspNet.Mvc4.nupkg
ASP.NET MVC5	Syncfusion.XlsIO.AspNet.Mvc5.nupkg
UWP, ASP.NET Core (Targeting .NET Core), Console Application (Targeting .NET Core)	[Syncfusion.XlsIORenderer.Net.Core.nupkg](https://www.nuget.org/packages/Syncfusion.XlsIORenderer.Net.Core/)
Xamarin	Syncfusion.Xamarin.XlsIORenderer.nupkg
Blazor (Server-Side)	{{ Syncfusion.XlsIORenderer.Net.Core.nupkg markdownify }}

Note: Worksheet to image conversion is supported from .NET Framework 2.0 and .NET Standard 1.4 onwards.

Converting Charts in XlsIO

The below NuGet package need to be installed additionally to convert the charts present in Excel documents.

Platform(s)	NuGet Package
Windows Forms,	Syncfusion.ExcelChartToImageConverter.WinForms.nupkg

Console Application (Targeting .NET Framework)	
WPF	Syncfusion.ExcelChartToImageConverter.Wpf.nupkg
ASP.NET Web Forms, ASP.NET Core (Targeting .NET Framework)	Syncfusion.ExcelChartToImageConverter.AspNet.nupkg
ASP.NET MVC4	Syncfusion.ExcelChartToImageConverter.AspNet.Mvc4.nupkg
ASP.NET MVC5	Syncfusion.ExcelChartToImageConverter.AspNet.Mvc5.nupkg
UWP, ASP.NET Core (Targeting .NET Core), Console Application (Targeting .NET Core)	[Syncfusion.XlsIORenderer.Net.Core.nupkg](https://www.nuget.org/packages/Syncfusion.XlsIORenderer.Net.Core/)
Xamarin	Syncfusion.Xamarin.XlsIORenderer.nupkg
Blazor (Server-Side)	{{' Syncfusion.XlsIORenderer.Net.Core.nupkg ' markdownify }}

Note: 1. The "Syncfusion.ExcelChartToImageConverter.Wpf.nupkg" NuGet package is only supported from 4.0 .NET Framework onwards.

2. The "Syncfusion.Xamarin.XlsIORenderer.nupkg" or "Syncfusion.XlsIORenderer.Net.Core.nupkg" NuGet packages supports chart to image conversion only from .NET Standard 2.0 onwards.

NuGet Package Installation and Uninstallation

To use NuGet package in your project, please refer the NuGet Package [Installation](#) and [Uninstallation](#) sections.

XlsIO NuGet packages can be installed and uninstalled using Package Manager Console. In Visual Studio, select **Tools > NuGet Package Manager > Package Manager Console** and execute the below commands in respective platforms.

Note: Syncfusion components are available in nuget.org

Platform(s)	Install	UnInstall
Windows Forms	<i>Install-package Syncfusion.XlsIO.WinForms</i> Install-package Syncfusion.ExcelToPdfConverter.WinForms * Install-package Syncfusion.ExcelChartToImageConverter.WinForms	<i>Uninstall-package Syncfusion.XlsIO.WinForms -RemoveDependencies</i> Uninstall-package Syncfusion.ExcelToPdfConverter.WinForms - RemoveDependencies * Uninstall-package Syncfusion.ExcelChartToImageConverter.WinForms -RemoveDependencies
WPF	<i>Install-package Syncfusion.XlsIO.Wpf</i> Install-package Syncfusion.ExcelToPdfConverter.Wpf * Install-package Syncfusion.ExcelChartToImageConverter.Wpf	<i>Uninstall-package Syncfusion.XlsIO.Wpf -RemoveDependencies</i> Uninstall-package Syncfusion.ExcelToPdfConverter.Wpf - RemoveDependencies * Uninstall-package Syncfusion.ExcelChartToImageConverter.Wpf - RemoveDependencies
ASP.NET Web Forms	<i>Install-package Syncfusion.XlsIO.AspNet</i> Install-package Syncfusion.ExcelToPdfConverter.AspNet * Install-package Syncfusion.ExcelChartToImageConverter.AspNet	<i>Uninstall-package Syncfusion.XlsIO.AspNet -RemoveDependencies</i> Uninstall-package Syncfusion.ExcelToPdfConverter.AspNet - RemoveDependencies * Uninstall-package Syncfusion.ExcelChartToImageConverter.AspNet -RemoveDependencies
ASP.NET MVC4	<i>Install-package Syncfusion.XlsIO.AspNet.MVC4</i> Install-package Syncfusion.ExcelToPdfConverter.AspNet.MVC4	<i>Uninstall-package Syncfusion.XlsIO.AspNet.MVC4 -RemoveDependencies</i> Uninstall-package

	* Install-package Syncfusion.ExcelChartToImageConverter.AspNet.MVC4	Syncfusion.ExcelToPdfConverter.AspNet.MVC4 -RemoveDependencies * Uninstall-package Syncfusion.ExcelChartToImageConverter.AspNet.MVC4 -RemoveDependencies
ASP.NET MVC5	<i>Install-package Syncfusion.XlsIO.AspNet.MVC5</i> Install-package Syncfusion.ExcelToPdfConverter.AspNet.MVC5 * Install-package Syncfusion.ExcelChartToImageConverter.AspNet.MVC5	<i>Uninstall-package Syncfusion.XlsIO.AspNet.MVC5 - RemoveDependencies</i> Uninstall-package Syncfusion.ExcelToPdfConverter.AspNet.MVC5 -RemoveDependencies * Uninstall-package Syncfusion.ExcelChartToImageConverter.AspNet.MVC5 -RemoveDependencies
UWP	Install-package Syncfusion.XlsIO.UWP	Uninstall-package Syncfusion.XlsIO.UWP â€”RemoveDependencies
ASP.NET Core and Blazor Server-Side	<i>Install-package Syncfusion.XlsIO.Net.Core</i> Install-package Syncfusion.XlsIORenderer.Net.Core	<i>Uninstall-package Syncfusion.XlsIO.Net.Core â€”RemoveDependencies</i> Uninstall-package Syncfusion.XlsIORenderer.Net.Core â€”RemoveDependencies
Xamarin	<i>Install-package Syncfusion.Xamarin.XlsIO</i> Install-package Syncfusion.Xamarin.XlsIORenderer	<i>Uninstall-package Syncfusion.Xamarin.XlsIO â€”RemoveDependencies</i> Uninstall-package Syncfusion.Xamarin.XlsIORenderer â€”RemoveDependencies
Blazor Client-Side	* Install-package Syncfusion.XlsIO.Net.Core	* Uninstall-package Syncfusion.XlsIO.Net.Core â€”RemoveDependencies

Getting Started - Create Excel File in C# and VB.NET

This section explains how to create a simple Excel file in C# and VB.NET using XlsIO. The following assemblies must be referred in your application to create and manipulate the Excel document.

Assembly Name	Description
Syncfusion.XlsIO.Base	This assembly contains the core features for creating, reading, and manipulating an Excel file.

Assembly Name	Description
Syncfusion.Compression.Base	This assembly is used to package the Workbook contents.

Note: Starting with v16.2.0.x, if you reference Syncfusion assemblies from trial setup or from the NuGet feed, you also have to add "Syncfusion.Licensing" assembly reference and include a license key in your projects. Please refer to this [link](#) to know about registering Syncfusion license key in your applications to use our components.

Note: Syncfusion components are available in nuget.org

Include the following namespaces in your .cs or .vb file as shown as follows.

C#

```
using Syncfusion.XlsIO;
```

VB.NET

```
Imports Syncfusion.XlsIO
```

UWP

```
using Syncfusion.XlsIO;
```

ASP.NET CORE

```
using Syncfusion.XlsIO;
```

XAMARIN

```
using Syncfusion.XlsIO;
```

Create a Hello World Excel File

The following code example explains how to create a hello world sample.

C#

```
using Syncfusion.XlsIO;
//New instance of ExcelEngine is created equivalent to launching Microsoft
Excel with no workbooks open
//Instantiate the spreadsheet creation engine
using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Instantiate the Excel application object
    IApplication application = excelEngine.Excel;
    //Assigns default application version
    application.DefaultVersion = ExcelVersion.Excel2013;
    //A new workbook is created equivalent to creating a new workbook in Excel
    //Create a workbook with 1 worksheet
    IWorkbook workbook = application.Workbooks.Create(1);
    //Access first worksheet from the workbook
```



```

IWorksheet worksheet = workbook.Worksheets[0];
//Adding text to a cell
worksheet.Range["A1"].Text = "Hello World";
//Saving the workbook to disk in XLSX format
workbook.SaveAs("Sample.xlsx");
}

```

VB.NET

```

Imports Syncfusion.XlsIO
'New instance of ExcelEngine is created equivalent to launching Microsoft
Excel with no workbooks open
'Instantiate the spreadsheet creation engine
Using excelEngine As ExcelEngine = New ExcelEngine()
'Instantiate the Excel application object
Dim application As IApplication = excelEngine.Excel
'Assigns default application version
application.DefaultVersion = ExcelVersion.Excel2013
'A new workbook is created equivalent to creating a new workbook in Excel
'Create a workbook with 1 worksheet
Dim workbook As IWorkbook = application.Workbooks.Create(1)
'Access first worksheet from workbook
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding text to a cell
worksheet.Range("A1").Text = "Hello World"
'Saving the workbook to disk in XLSX format
workbook.SaveAs("Sample.xlsx")
End Using

```

UWP

```

using Syncfusion.XlsIO;
//New instance of ExcelEngine is created equivalent to launching Microsoft
Excel with no workbooks open
//Instantiate the spreadsheet creation engine
using (ExcelEngine excelEngine = new ExcelEngine())
{
//Instantiate the Excel application object
IApplication application = excelEngine.Excel;
//Assigns default application version
application.DefaultVersion = ExcelVersion.Excel2013;
//A new workbook is created equivalent to creating a new workbook in Excel
//Create a workbook with 1 worksheet
IWorkbook workbook = application.Workbooks.Create(1);
//Access first worksheet from the workbook
IWorksheet worksheet = workbook.Worksheets[0];
//Adding text to a cell
worksheet.Range["A1"].Text = "Hello World";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
}

```

```
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

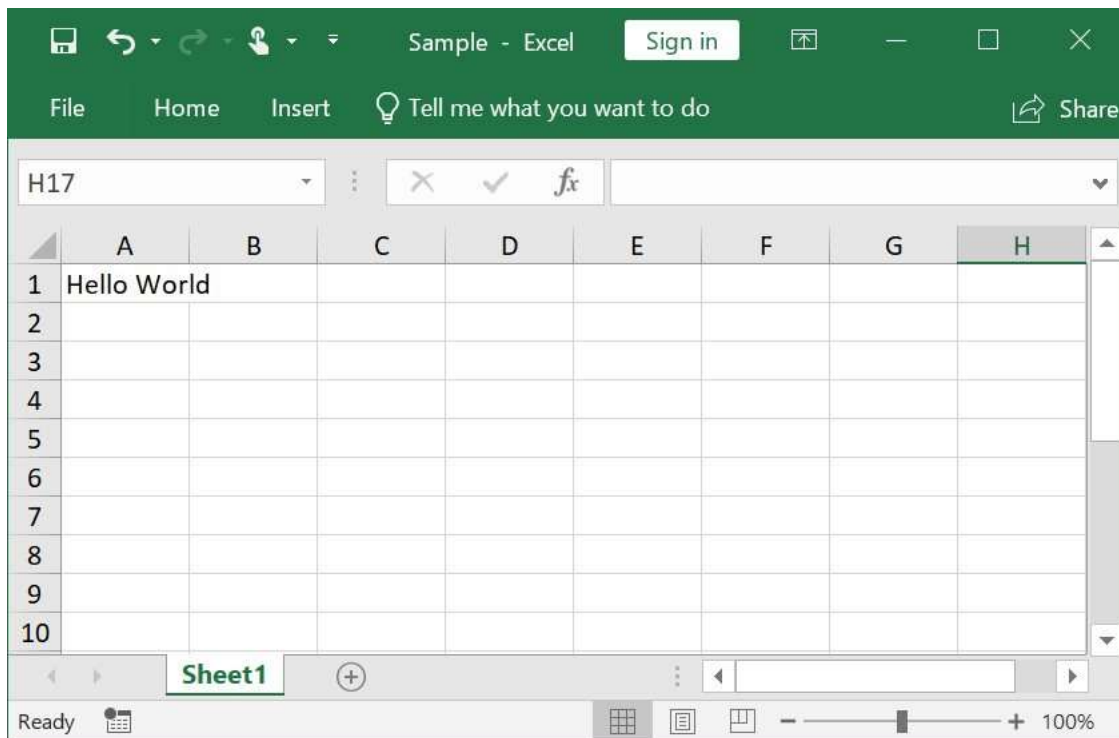
```
using Syncfusion.XlsIO;
//New instance of ExcelEngine is created equivalent to launching Microsoft
Excel with no workbooks open
//Instantiate the spreadsheet creation engine
using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Instantiate the Excel application object
    IApplication application = excelEngine.Excel;
    //Assigns default application version
    application.DefaultVersion = ExcelVersion.Excel2013;
    //A new workbook is created equivalent to creating a new workbook in Excel
    //Create a workbook with 1 worksheet
    IWorkbook workbook = application.Workbooks.Create(1);
    //Access first worksheet from the workbook
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding text to a cell
    worksheet.Range["A1"].Text = "Hello World";
    //Saving the workbook as stream
    FileStream stream = new FileStream("Sample.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    //Dispose stream
    stream.Dispose();
}
```

XAMARIN

```
using Syncfusion.XlsIO;
//New instance of ExcelEngine is created equivalent to launching Microsoft
Excel with no workbooks open
//Instantiate the spreadsheet creation engine
using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Instantiate the Excel application object
    IApplication application = excelEngine.Excel;
    //Assigns default application version
    application.DefaultVersion = ExcelVersion.Excel2013;
    //A new workbook is created equivalent to creating a new workbook in Excel
    //Create a workbook with 1 worksheet
    IWorkbook workbook = application.Workbooks.Create(1);
    //Access first worksheet from the workbook
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding text to a cell
    worksheet.Range["A1"].Text = "Hello World";
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
```

```
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Sample
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.xlsx",
"application/msexcel", stream);
}
}
```

The output screen-shot of the above code.



Create a Simple Excel File

An instance of the `ExcelEngine` gives access to create an application instance that is similar to launching Microsoft Excel application. The following code snippet shows how to initialize the application object for creating or manipulating Excel documents.

C#

```
//New instance of ExcelEngine is created equivalent to launching Microsoft
//Excel with no workbooks open
//Instantiate the spreadsheet creation engine
ExcelEngine excelEngine = new ExcelEngine();
//Instantiate the Excel application object
IApplication application = excelEngine.Excel;
```

VB.NET

```
'New instance of ExcelEngine is created equivalent to launching Microsoft
Excel with no workbooks open
'Instantiate the spreadsheet creation engine
Dim excelEngine As ExcelEngine = New ExcelEngine
'Instantiate the Excel application object
Dim application As IApplication = excelEngine.Excel
```

UWP

```
//New instance of ExcelEngine is created equivalent to launching Microsoft
Excel with no workbooks open
//Instantiate the spreadsheet creation engine
ExcelEngine excelEngine = new ExcelEngine();
//Instantiate the Excel application object
IApplication application = excelEngine.Excel;
```

ASP.NET CORE

```
//New instance of ExcelEngine is created equivalent to launching Microsoft
Excel with no workbooks open
//Instantiate the spreadsheet creation engine
ExcelEngine excelEngine = new ExcelEngine();
//Instantiate the Excel application object
IApplication application = excelEngine.Excel;
```

XAMARIN

```
//New instance of ExcelEngine is created equivalent to launching Microsoft
Excel with no workbooks open
//Instantiate the spreadsheet creation engine
ExcelEngine excelEngine = new ExcelEngine();
//Instantiate the Excel application object
IApplication application = excelEngine.Excel;
```

By default, the Excel version 97 to 2003 (*.xls) is associated with application object. XlsIO writes the excel files in the respective format depending on this excel version. You can modify the default Excel version to Excel 2013 as shown as follows.

C#

```
//Assigns default application version
application.DefaultVersion = ExcelVersion.Excel2013;
```

VB.NET

```
'Assigns default application version
application.DefaultVersion = ExcelVersion.Excel2013
```

UWP

```
//Assigns default application version
application.DefaultVersion = ExcelVersion.Excel2013;
```

ASP.NET CORE

```
//Assigns default application version
application.DefaultVersion = ExcelVersion.Excel2013;
```

XAMARIN

```
//Assigns default application version
application.DefaultVersion = ExcelVersion.Excel2013;
```

The workbook contains a collection of worksheets and various workbook-level properties. Each worksheet has cells, which can contain text, numbers, dates, formulas and more. The following code snippet illustrates how to create a workbook and access worksheet instance.

C#

```
//A new workbook is created equivalent to creating a new workbook in Excel
//Create a workbook with 1 worksheet
IWorkbook workbook = application.Workbooks.Create(1);
//Access a worksheet from workbook
IWorksheet worksheet = workbook.Worksheets[0];
```

VB.NET

```
'A new workbook is created equivalent to creating a new workbook in Excel
'Create a workbook with 1 worksheet
Dim workbook As IWorkbook = application.Workbooks.Create(1)
'Access a worksheet from workbook
Dim worksheet As IWorksheet = workbook.Worksheets(0)
```

UWP

```
//A new workbook is created equivalent to creating a new workbook in Excel
//Create a workbook with 1 worksheet
IWorkbook workbook = application.Workbooks.Create(1);
//Access a worksheet from workbook
IWorksheet worksheet = workbook.Worksheets[0];
```

ASP.NET CORE

```
//A new workbook is created equivalent to creating a new workbook in Excel
//Create a workbook with 1 worksheet
IWorkbook workbook = application.Workbooks.Create(1);
//Access a worksheet from workbook
IWorksheet worksheet = workbook.Worksheets[0];
```

XAMARIN

```
//A new workbook is created equivalent to creating a new workbook in Excel
//Create a workbook with 1 worksheet
```

```
IWorkbook workbook = application.Workbooks.Create(1);
//Access a worksheet from workbook
IWorksheet worksheet = workbook.Worksheets[0];
```

C#

```
//Adding text data
worksheet.Range["A1"].Text = "Month";
worksheet.Range["B1"].Text = "Sales";
worksheet.Range["A6"].Text = "Total";
//Adding DateTime data
worksheet.Range["A2"].DateTime = new DateTime(2015, 1, 10);
worksheet.Range["A3"].DateTime = new DateTime(2015, 2, 10);
worksheet.Range["A4"].DateTime = new DateTime(2015, 3, 10);
//Applying number format for date value cells A2 to A4
worksheet.Range["A2:A4"].NumberFormat = "mmmm, yyyy";
//Auto-size the first column to fit the content
worksheet.AutofitColumn(1);
//Adding numeric data
worksheet.Range["B2"].Number = 68878;
worksheet.Range["B3"].Number = 71550;
worksheet.Range["B4"].Number = 72808;
//Adding formula
worksheet.Range["B6"].Formula = "SUM(B2:B4)";
```

VB.NET

```
'Adding text data
worksheet.Range("A1").Text = "Month"
worksheet.Range("B1").Text = "Sales"
worksheet.Range("A6").Text = "Total"
'Adding DateTime data
worksheet.Range("A2").DateTime = new DateTime(2015, 1, 10)
worksheet.Range("A3").DateTime = new DateTime(2015, 2, 10)
worksheet.Range("A4").DateTime = new DateTime(2015, 3, 10)
'Applying number format for date value cells A2 to A4
worksheet.Range("A2:A4").NumberFormat = "mmmm, yyyy"
'Auto-size the first column to fit the content
worksheet.AutofitColumn(1)
'Adding numeric data
worksheet.Range("B2").Number = 68878
worksheet.Range("B3").Number = 71550
worksheet.Range("B4").Number = 72808
'Adding formula
worksheet.Range("B6").Formula = "SUM(B2:B4) "
```

UWP

```
//Adding text data
worksheet.Range["A1"].Text = "Month";
worksheet.Range["B1"].Text = "Sales";
worksheet.Range["A6"].Text = "Total";
//Adding DateTime data
worksheet.Range["A2"].DateTime = new DateTime(2015, 1, 10);
worksheet.Range["A3"].DateTime = new DateTime(2015, 2, 10);
```

```

worksheet.Range["A4"].DateTime = new DateTime(2015, 3, 10);
//Applying number format for date value cells A2 to A4
worksheet.Range["A2:A4"].NumberFormat = "mmmm, yyyy";
//Auto-size the first column to fit the content
worksheet.AutofitColumn(1);
//Adding numeric data
worksheet.Range["B2"].Number = 68878;
worksheet.Range["B3"].Number = 71550;
worksheet.Range["B4"].Number = 72808;
//Adding formula
worksheet.Range["B6"].Formula = "SUM(B2:B4)";

```

ASP.NET CORE

```

//Adding text data
worksheet.Range["A1"].Text = "Month";
worksheet.Range["B1"].Text = "Sales";
worksheet.Range["A6"].Text = "Total";
//Adding DateTime data
worksheet.Range["A2"].DateTime = new DateTime(2015, 1, 10);
worksheet.Range["A3"].DateTime = new DateTime(2015, 2, 10);
worksheet.Range["A4"].DateTime = new DateTime(2015, 3, 10);
//Applying number format for date value cells A2 to A4
worksheet.Range["A2:A4"].NumberFormat = "mmmm, yyyy";
//Auto-size the first column to fit the content
worksheet.AutofitColumn(1);
//Adding numeric data
worksheet.Range["B2"].Number = 68878;
worksheet.Range["B3"].Number = 71550;
worksheet.Range["B4"].Number = 72808;
//Adding formula
worksheet.Range["B6"].Formula = "SUM(B2:B4)";

```

XAMARIN

```

//Adding text data
worksheet.Range["A1"].Text = "Month";
worksheet.Range["B1"].Text = "Sales";
worksheet.Range["A6"].Text = "Total";
//Adding DateTime data
worksheet.Range["A2"].DateTime = new DateTime(2015, 1, 10);
worksheet.Range["A3"].DateTime = new DateTime(2015, 2, 10);
worksheet.Range["A4"].DateTime = new DateTime(2015, 3, 10);
//Applying number format for date value cells A2 to A4
worksheet.Range["A2:A4"].NumberFormat = "mmmm, yyyy";
//Auto-size the first column to fit the content
worksheet.AutofitColumn(1);
//Adding numeric data
worksheet.Range["B2"].Number = 68878;
worksheet.Range["B3"].Number = 71550;
worksheet.Range["B4"].Number = 72808;
//Adding formula
worksheet.Range["B6"].Formula = "SUM(B2:B4)";

```

The following code snippet shows how to add an image into the worksheet.

C#

```
//Inserting image  
worksheet.Pictures.AddPicture(10, 2, "image.jpg");
```

VB.NET

```
'Inserting image  
worksheet.Pictures.AddPicture(10, 2, "image.jpg")
```

UWP

```
//Inserting image  
//"App" is the class of Portable project  
Assembly assembly = typeof(App).GetTypeInfo().Assembly;  
Stream imageStream =  
assembly.GetManifestResourceStream("UWP.Data.image.jpg");  
worksheet.Pictures.AddPicture(10, 2, imageStream);
```

ASP.NET CORE

```
//Inserting image  
FileStream imageStream = new FileStream("image.jpg", FileMode.Open,  
FileAccess.Read);  
worksheet.Pictures.AddPicture(10, 2, imageStream);
```

XAMARIN

```
//Inserting image  
//"App" is the class of Portable project  
Assembly assembly = typeof(App).GetTypeInfo().Assembly;  
Stream imageStream =  
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.image.jpg");  
worksheet.Pictures.AddPicture(10, 2, imageStream);
```

Finally, save the document in file system and close/dispose the instance of IWorkbook and ExcelEngine.

C#

```
//Saving the workbook to disk in XLSX format  
workbook.SaveAs("Sample.xlsx");  
//Closing the workbook  
workbook.Close();  
//Dispose the Excel engine  
excelEngine.Dispose();
```

VB.NET

```
'Saving the workbook to disk in XLSX format  
workbook.SaveAs("Sample.xlsx")  
'Closing the workbook  
workbook.Close()  
'Dispose the Excel engine
```



```
excelEngine.Dispose()
```

UWP

```
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
//Closing the workbook
workbook.Close();
//Dispose the Excel engine
excelEngine.Dispose();
```

ASP.NET CORE

```
//Save the workbook as stream
FileStream stream = new FileStream("Sample.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
//Disposing the stream
stream.Dispose();
//Closing the workbook
workbook.Close();
//Dispose the Excel engine
excelEngine.Dispose();
```

XAMARIN

```
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
//Closing the workbook
workbook.Close();
//Dispose the Excel engine
excelEngine.Dispose();
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Sample
.xlsx", "application/msexcel", stream);
}
else
{
```

```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.xlsx",  
"application/msexcel", stream);  
}
```

The complete code to create a simple Excel document.

C#

```
using Syncfusion.XlsIO;  
namespace ExcelCreation  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            //New instance of ExcelEngine is created equivalent to launching Excel with  
            //no workbooks open  
            //Instantiate the spreadsheet creation engine  
            using (ExcelEngine excelEngine = new ExcelEngine())  
            {  
                //Instantiate the Excel application object  
                IApplication application = excelEngine.Excel;  
                //Assigns default application version  
                application.DefaultVersion = ExcelVersion.Excel2013;  
                //A new workbook is created equivalent to creating a new workbook in Excel  
                //Create a workbook with 1 worksheet  
                IWorkbook workbook = application.Workbooks.Create(1);  
                //Access a worksheet from workbook  
                IWorksheet worksheet = workbook.Worksheets[0];  
                //Adding text data  
                worksheet.Range["A1"].Text = "Month";  
                worksheet.Range["B1"].Text = "Sales";  
                worksheet.Range["A6"].Text = "Total";  
                //Adding DateTime data  
                worksheet.Range["A2"].DateTime = new DateTime(2015, 1, 10);  
                worksheet.Range["A3"].DateTime = new DateTime(2015, 2, 10);  
                worksheet.Range["A4"].DateTime = new DateTime(2015, 3, 10);  
                //Applying number format for date value cells A2 to A4  
                worksheet.Range["A2:A4"].NumberFormat = "mmmm, yyyy";  
                //Auto-size the first column to fit the content  
                worksheet.AutofitColumn(1);  
                //Adding numeric data  
                worksheet.Range["B2"].Number = 68878;  
                worksheet.Range["B3"].Number = 71550;  
                worksheet.Range["B4"].Number = 72808;  
                //Adding formula  
                worksheet.Range["B6"].Formula = "SUM(B2:B4)";  
                //Inserting image  
                worksheet.Pictures.AddPicture(10, 2, "image.jpg");  
                //Saving the workbook to disk in XLSX format  
                workbook.SaveAs("Sample.xlsx");  
            }  
        }  
    }  
}
```

VB.NET

```
Imports Syncfusion.XlsIO
Namespace ExcelCreation
Module Program
Sub Main(args As String())
    'New instance of ExcelEngine is created equivalent to launching Microsoft
    'Excel with no workbooks open
    'Instantiate the spreadsheet creation engine
    Using excelEngine As ExcelEngine = New ExcelEngine()
        'Instantiate the Excel application object
        Dim application As IApplication = excelEngine.Excel
        'Assigns default application version
        application.DefaultVersion = ExcelVersion.Excel2013
        'A new workbook is created equivalent to creating a new workbook in Excel
        'Create a workbook with 1 worksheet
        Dim workbook As IWorkbook = application.Workbooks.Create(1)
        'Access a worksheet from workbook
        Dim worksheet As IWorksheet = workbook.Worksheets(0)
        'Adding text data
        worksheet.Range("A1").Text = "Month"
        worksheet.Range("B1").Text = "Sales"
        worksheet.Range("A6").Text = "Total"
        'Adding DateTime data
        worksheet.Range("A2").DateTime = New DateTime(2015, 1, 10)
        worksheet.Range("A3").DateTime = New DateTime(2015, 2, 10)
        worksheet.Range("A4").DateTime = New DateTime(2015, 3, 10)
        'Applying number format for date value cells A2 to A4
        worksheet.Range("A2:A4").NumberFormat = "mmmm, yyyy"
        'Auto-size the first column to fit the content
        worksheet.AutofitColumn(1)
        'Adding numeric data
        worksheet.Range("B2").Number = 68878
        worksheet.Range("B3").Number = 71550
        worksheet.Range("B4").Number = 72808
        'Adding formula
        worksheet.Range("B6").Formula = "SUM(B2:B4)"
        'Inserting image
        worksheet.Pictures.AddPicture(10, 2, "image.jpg")
        'Saving the workbook to disk in XLSX format
        workbook.SaveAs("Sample.xlsx")
    End Using
End Sub
End Module
End Namespace
```

UWP

```
using Syncfusion.XlsIO;
namespace ExcelCreation
{
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }
    }
}
```

```

}
private async void OnButtonClicked(object sender, RoutedEventArgs e)
{
    //New instance of ExcelEngine is created equivalent to launching Excel with
    //no workbooks open
    //Instantiate the spreadsheet creation engine
    using (ExcelEngine excelEngine = new ExcelEngine())
    {
        //Instantiate the Excel application object
        IApplication application = excelEngine.Excel;
        //Assigns default application version
        application.DefaultVersion = ExcelVersion.Excel2013;
        //A new workbook is created equivalent to creating a new workbook in Excel
        //Create a workbook with 1 worksheet
        IWorkbook workbook = application.Workbooks.Create(1);
        //Access a worksheet from workbook
        IWorksheet worksheet = workbook.Worksheets[0];
        //Adding text data
        worksheet.Range["A1"].Text = "Month";
        worksheet.Range["B1"].Text = "Sales";
        worksheet.Range["A6"].Text = "Total";
        //Adding DateTime data
        worksheet.Range["A2"].DateTime = new DateTime(2015, 1, 10);
        worksheet.Range["A3"].DateTime = new DateTime(2015, 2, 10);
        worksheet.Range["A4"].DateTime = new DateTime(2015, 3, 10);
        //Applying number format for date value cells A2 to A4
        worksheet.Range["A2:A4"].NumberFormat = "mmmm, yyyy";
        //Auto-size the first column to fit the content
        worksheet.AutofitColumn(1);
        //Adding numeric data
        worksheet.Range["B2"].Number = 68878;
        worksheet.Range["B3"].Number = 71550;
        worksheet.Range["B4"].Number = 72808;
        //Adding formula
        worksheet.Range["B6"].Formula = "SUM(B2:B4)";
        //Inserting image
        //"App" is the class of Portable project
        Assembly assembly = typeof(App).GetTypeInfo().Assembly;
        Stream imageStream =
            assembly.GetManifestResourceStream("UWP.Data.image.jpg");
        worksheet.Pictures.AddPicture(10, 2, imageStream);
        //Initializes FileSavePicker
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
        savePicker.SuggestedFileName = "Sample";
        savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
        });
        //Creates a storage file from FileSavePicker
        StorageFile storageFile = await savePicker.PickSaveFileAsync();
        //Saves changes to the specified storage file
        await workbook.SaveAsAsync(storageFile);
    }
}
}
}
}

```

ASP.NET CORE

```
using Syncfusion.XlsIO;
namespace ExcelCreation
{
    class Program
    {
        static void Main(string[] args)
        {
            //New instance of ExcelEngine is created equivalent to launching Excel with
            no workbooks open
            //Instantiate the spreadsheet creation engine
            using (ExcelEngine excelEngine = new ExcelEngine())
            {
                //Instantiate the Excel application object
                IApplication application = excelEngine.Excel;
                //Assigns default application version
                application.DefaultVersion = ExcelVersion.Excel2013;
                //A new workbook is created equivalent to creating a new workbook in Excel
                //Create a workbook with 1 worksheet
                IWorkbook workbook = application.Workbooks.Create(1);
                //Access a worksheet from workbook
                IWorksheet worksheet = workbook.Worksheets[0];
                //Adding text data
                worksheet.Range["A1"].Text = "Month";
                worksheet.Range["B1"].Text = "Sales";
                worksheet.Range["A6"].Text = "Total";
                //Adding DateTime data
                worksheet.Range["A2"].DateTime = new DateTime(2015, 1, 10);
                worksheet.Range["A3"].DateTime = new DateTime(2015, 2, 10);
                worksheet.Range["A4"].DateTime = new DateTime(2015, 3, 10);
                //Applying number format for date value cells A2 to A4
                worksheet.Range["A2:A4"].NumberFormat = "mmmm, yyyy";
                //Auto-size the first column to fit the content
                worksheet.AutofitColumn(1);
                //Adding numeric data
                worksheet.Range["B2"].Number = 68878;
                worksheet.Range["B3"].Number = 71550;
                worksheet.Range["B4"].Number = 72808;
                //Adding formula
                worksheet.Range["B6"].Formula = "SUM(B2:B4)";
                //Inserting image
                FileStream imageStream = new FileStream("image.jpg", FileMode.Open,
                FileAccess.Read);
                worksheet.Pictures.AddPicture(10, 2, imageStream);
                //Saving the workbook to disk in XLSX format
                FileStream stream = new FileStream("Sample.xlsx", FileMode.Create,
                FileAccess.ReadWrite);
                workbook.SaveAs(stream);
                //Dispose stream
                stream.Dispose();
            }
        }
    }
}
```

XAMARIN

```

using Syncfusion.XlsIO;
namespace ExcelCreation
{
    class Program
    {
        static void Main(string[] args)
        {
            //New instance of ExcelEngine is created equivalent to launching Excel with
            //no workbooks open
            //Instantiate the spreadsheet creation engine
            using (ExcelEngine excelEngine = new ExcelEngine())
            {
                //Instantiate the Excel application object
                IApplication application = excelEngine.Excel;
                //Assigns default application version
                application.DefaultVersion = ExcelVersion.Excel2013;
                //A new workbook is created equivalent to creating a new workbook in Excel
                //Create a workbook with 1 worksheet
                IWorkbook workbook = application.Workbooks.Create(1);
                //Access a worksheet from workbook
                IWorksheet worksheet = workbook.Worksheets[0];
                //Adding text data
                worksheet.Range["A1"].Text = "Month";
                worksheet.Range["B1"].Text = "Sales";
                worksheet.Range["A6"].Text = "Total";
                //Adding DateTime data
                worksheet.Range["A2"].DateTime = new DateTime(2015, 1, 10);
                worksheet.Range["A3"].DateTime = new DateTime(2015, 2, 10);
                worksheet.Range["A4"].DateTime = new DateTime(2015, 3, 10);
                //Applying number format for date value cells A2 to A4
                worksheet.Range["A2:A4"].NumberFormat = "mmmm, yyyy";
                //Auto-size the first column to fit the content
                worksheet.AutofitColumn(1);
                //Adding numeric data
                worksheet.Range["B2"].Number = 68878;
                worksheet.Range["B3"].Number = 71550;
                worksheet.Range["B4"].Number = 72808;
                //Adding formula
                worksheet.Range["B6"].Formula = "SUM(B2:B4)";
                //Inserting image
                //App is the class of Portable project
                Assembly assembly = typeof(App).GetTypeInfo().Assembly;
                Stream imageStream =
                    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.ima
                    ge.jpg");
                worksheet.Pictures.AddPicture(10, 2, imageStream);
                //Saving the workbook as stream
                MemoryStream stream = new MemoryStream();
                workbook.SaveAs(stream);
                stream.Position = 0;
                //Save the document as file and view the saved document
                //The operation in SaveAndView under Xamarin varies between Windows Phone,
                //Android and iOS platforms. Please refer xlsio/xamarin section for respective
                //code samples.
            }
        }
    }
}

```

```
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==  
TargetPlatform.Windows)  
{  
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Sample  
.xlsx", "application/msexcel", stream);  
}  
else  
{  
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.xlsx",  
"application/msexcel", stream);  
}  
}  
}  
}
```

The output screen-shot of the above code.



Import Data to Excel Worksheets

XlsIO helps to import data from various data sources into a worksheet. The following data sources can be imported using XlsIO:

- Collection Objects
- Data Table
- Data Column
- Data View

- Array

The following code snippet shows how to import data from objects.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //GetCustomerAsObjects method returns list of customers
    IList<Employee> employees = GetEmployees();
    //Import data to worksheet
    worksheet.ImportData(employees, 2, 1, false);
    //Saving the workbook
    workbook.SaveAs("Spreadsheet.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'GetCustomerAsObjects method returns list of customers
Dim employees As IList(Of Employee) = GetEmployees()
'Import data to worksheet
worksheet.ImportData(employees, 2, 1, False)
'Saving the workbook
workbook.SaveAs("Spreadsheet.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //GetCustomerAsObjects method returns list of customers
    IList<Employee> employees = GetEmployees();
    //Import data to worksheet
    worksheet.ImportData(employees, 2, 1, false);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Spreadsheet";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx" });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
```



```
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //GetCustomerAsObjects method returns list of customers
    IList<Employee> employees = GetEmployees();
    //Import data to worksheet
    worksheet.ImportData(employees, 2, 1, false);
    //Saving the workbook as stream
    FileStream file = new FileStream("Spreadsheet.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(file);
    file.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //GetCustomerAsObjects method returns list of customers
    IList<Employee> employees = GetEmployees();
    //Import data to worksheet
    worksheet.ImportData(employees, 2, 1, false);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Spread
        sheet.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Spreadsheet.xlsx",
        "application/msexcel", stream);
    }
}
```

The following code snippet provides supporting methods and classes for the previous code.

C#

```
//Gets a list of Employee details
private static IList<Employee> GetEmployees()
{
    List<Employee> employees = new List<Employee>();
    employees.Add(new Employee("Nancy", "Davolio", "Sales Representative", "505
- 20th Ave. E. Apt. 2A,", "Seattle", "WA", "USA", "Nancy.png"));
    employees.Add(new Employee("Andrew", "Fuller", "Vice President, Sales", "908
W. Capital Way", "Tacoma", "WA", "USA", "Andrew.png"));
    employees.Add(new Employee("Janet", "Leverling", "Sales Representative",
"722 Moss Bay Blvd.", "Kirkland", "WA", "USA", "Janet.png"));
    employees.Add(new Employee("Margaret", "Peacock", "Sales Representative",
"4110 Old Redmond Rd.", "Redmond", "WA", "USA", "Margaret.png"));
    employees.Add(new Employee("Steven", "Buchanan", "Sales Manager", "14
Garrett Hill", "London", string.Empty, "UK", "Steven.png"));
    return employees;
}

//Employee details
public class Employee
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Address { get; set; }
    public string City { get; set; }
    public string Region { get; set; }
    public string Country { get; set; }
    public string Title { get; set; }
    public Employee(string firstName, string lastName, string title, string
address, string city, string region, string country, string photoFilePath)
    {
        FirstName = firstName;
        LastName = lastName;
        Title = title;
        Address = address;
        City = city;
        Region = region;
        Country = country;
    }
}
```

VB.NET

```
'Gets a list Employee details
Private Function GetEmployees() As List(Of Employee)
Dim employees As New List(Of Employee)()
employees.Add(New Employee("Nancy", "Davolio", "Sales Representative", "505
- 20th Ave. E. Apt. 2A,", "Seattle", "WA", "USA", "Nancy.png"))
employees.Add(New Employee("Andrew", "Fuller", "Vice President, Sales", "908
W. Capital Way", "Tacoma", "WA", "USA", "Andrew.png"))
employees.Add(New Employee("Janet", "Leverling", "Sales Representative",
"722 Moss Bay Blvd.", "Kirkland", "WA", "USA", "Janet.png"))
employees.Add(New Employee("Margaret", "Peacock", "Sales Representative",
"4110 Old Redmond Rd.", "Redmond", "WA", "USA", "Margaret.png"))
```

```
employees.Add(New Employee("Steven", "Buchanan", "Sales Manager", "14  
Garrett Hill", "London", String.Empty, "UK", "Steven.png"))  
Return employees  
End Function  
  
'Employee details  
Public Class Employee  
Public Property FirstName() As String  
Get  
Return m_FirstName  
End Get  
Set(value As String)  
m_FirstName = Value  
End Set  
End Property  
Private m_FirstName As String  
Public Property LastName() As String  
Get  
Return m_LastName  
End Get  
Set(value As String)  
m_LastName = Value  
End Set  
End Property  
Private m_LastName As String  
Public Property Address() As String  
Get  
Return m_Address  
End Get  
Set(value As String)  
m_Address = Value  
End Set  
End Property  
Private m_Address As String  
Public Property City() As String  
Get  
Return m_City  
End Get  
Set(value As String)  
m_City = Value  
End Set  
End Property  
Private m_City As String  
Public Property Region() As String  
Get  
Return m_Region  
End Get  
Set(value As String)  
m_Region = Value  
End Set  
End Property  
Private m_Region As String  
Public Property Country() As String  
Get  
Return m_Country  
End Get  
Set(value As String)  
m_Country = Value
```

```

End Set
End Property
Private m_Country As String
Public Property Title() As String
Get
Return m_Title
End Get
Set(value As String)
m_Title = Value
End Set
End Property
Private m_Title As String
Public Sub New(firstName As String, lastName As String, title As String,
address As String, city As String, region As String, country As String,
photoFilePath As String)
firstName = firstName
lastName = lastName
title = title
address = address
city = city
region = region
country = country
End Sub
End Class

```

UWP

```

//Gets a list of Employee details
private static List<Employee> GetEmployees()
{
List<Employee> employees = new List<Employee>();
employees.Add(new Employee("Nancy", "Davolio", "Sales Representative", "505
- 20th Ave. E. Apt. 2A,", "Seattle", "WA", "USA", "Nancy.png"));
employees.Add(new Employee("Andrew", "Fuller", "Vice President, Sales", "908
W. Capital Way", "Tacoma", "WA", "USA", "Andrew.png"));
employees.Add(new Employee("Janet", "Leverling", "Sales Representative",
"722 Moss Bay Blvd.", "Kirkland", "WA", "USA", "Janet.png"));
employees.Add(new Employee("Margaret", "Peacock", "Sales Representative",
"4110 Old Redmond Rd.", "Redmond", "WA", "USA", "Margaret.png"));
employees.Add(new Employee("Steven", "Buchanan", "Sales Manager", "14
Garrett Hill", "London", string.Empty, "UK", "Steven.png"));
return employees;
}
//Employee details
public class Employee
{
public string FirstName { get; set; }
public string LastName { get; set; }
public string Address { get; set; }
public string City { get; set; }
public string Region { get; set; }
public string Country { get; set; }
public string Title { get; set; }
public Employee(string firstName, string lastName, string title, string
address, string city, string region, string country, string photoFilePath)
{

```

```

FirstName = firstName;
LastName = lastName;
Title = title;
Address = address;
City = city;
Region = region;
Country = country;
}
}

```

ASP.NET CORE

```

//Gets a list of Employee details
private static List<Employee> GetEmployees()
{
    List<Employee> employees = new List<Employee>();
    employees.Add(new Employee("Nancy", "Davolio", "Sales Representative", "505
- 20th Ave. E. Apt. 2A,", "Seattle", "WA", "USA", "Nancy.png"));
    employees.Add(new Employee("Andrew", "Fuller", "Vice President, Sales", "908
W. Capital Way", "Tacoma", "WA", "USA", "Andrew.png"));
    employees.Add(new Employee("Janet", "Leverling", "Sales Representative",
"722 Moss Bay Blvd.", "Kirkland", "WA", "USA", "Janet.png"));
    employees.Add(new Employee("Margaret", "Peacock", "Sales Representative",
"4110 Old Redmond Rd.", "Redmond", "WA", "USA", "Margaret.png"));
    employees.Add(new Employee("Steven", "Buchanan", "Sales Manager", "14
Garrett Hill", "London", string.Empty, "UK", "Steven.png"));
    return employees;
}
//Employee details
public class Employee
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Address { get; set; }
    public string City { get; set; }
    public string Region { get; set; }
    public string Country { get; set; }
    public string Title { get; set; }
    public Employee(string firstName, string lastName, string title, string
address, string city, string region, string country, string photoFilePath)
    {
        FirstName = firstName;
        LastName = lastName;
        Title = title;
        Address = address;
        City = city;
        Region = region;
        Country = country;
    }
}

```

XAMARIN

```

//Gets a list of Employee details
private static List<Employee> GetEmployees()
{

```

```

List<Employee> employees = new List<Employee>();
employees.Add(new Employee("Nancy", "Davolio", "Sales Representative", "505
- 20th Ave. E. Apt. 2A,", "Seattle", "WA", "USA", "Nancy.png"));
employees.Add(new Employee("Andrew", "Fuller", "Vice President, Sales", "908
W. Capital Way", "Tacoma", "WA", "USA", "Andrew.png"));
employees.Add(new Employee("Janet", "Leverling", "Sales Representative",
"722 Moss Bay Blvd.", "Kirkland", "WA", "USA", "Janet.png"));
employees.Add(new Employee("Margaret", "Peacock", "Sales Representative",
"4110 Old Redmond Rd.", "Redmond", "WA", "USA", "Margaret.png"));
employees.Add(new Employee("Steven", "Buchanan", "Sales Manager", "14
Garrett Hill", "London", string.Empty, "UK", "Steven.png"));
return employees;
}
//Employee details
public class Employee
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Address { get; set; }
    public string City { get; set; }
    public string Region { get; set; }
    public string Country { get; set; }
    public string Title { get; set; }
    public Employee(string firstName, string lastName, string title, string
address, string city, string region, string country, string photoFilePath)
    {
        FirstName = firstName;
        LastName = lastName;
        Title = title;
        Address = address;
        City = city;
        Region = region;
        Country = country;
    }
}

```

You can refer various importing options in the “Importing Data to Worksheet” section.

Export Data from Excel Worksheets

The worksheet data can be exported to a data table using the **ExportDataTable()** method. This method provides various options that allows to export data through ExcelExportDataTableOptions.

The following code demonstrates how to export data from a worksheet to a data table with the **ColumnNames** and **DetectColumnTypes** options.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("WorkbookWithData.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Export data from worksheet used range to a DataTable
}

```

```

DataTable customersTable = sheet.ExportDataTable(sheet.UsedRange,
ExcelExportDataTableOptions.ColumnNames |
ExcelExportDataTableOptions.DetectColumnTypes);
//Saving the workbook
string fileName = "Output.xlsx";
workbook.SaveAs(fileName);
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook =
application.Workbooks.Open("WorkbookWithData.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
application.DefaultVersion = ExcelVersion.Excel2013
'Export data from worksheet used range to a DataTable
Dim customersTable As DataTable = sheet.ExportDataTable(sheet.UsedRange,
ExcelExportDataTableOptions.ColumnNames Or
ExcelExportDataTableOptions.DetectColumnTypes)
'Saving the workbook
Dim fileName As String = "Output.xlsx"
workbook.SaveAs(fileName)
End Using

```

UWP

```

//XlsIO supports exporting of data from worksheet to data table from .NET
Standard 2.0 along with Windows Forms, WPF, ASP.NET and ASP.NET MVC
platforms alone.
//Exporting data from worksheet can be achieved using List as illustrated
below.
//To know more about exporting data from worksheet to various collection
objects, please refer xlsio/working-with-data section.
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile openFile = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(openFile);
IWorksheet worksheet = workbook.Worksheets[0];
//Export data
List<Sales> data = worksheet.ExportData<Sales>(1, 1, 41, 4);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "CreateSpreadsheet";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker

```

```
StorageFile storageFile = await savePicker.PickSaveFileAsync();  
//Saves changes to the specified storage file  
await workbook.SaveAsAsync(storageFile);  
}  
//Sales details  
public class Sales  
{  
    private string salesPerson;  
    private int salesJanJune;  
    private int salesJulyDec;  
    private int change;  
    public string SalesPerson  
    {  
        get  
        {  
            return salesPerson;  
        }  
        set  
        {  
            salesPerson = value;  
        }  
    }  
    public int SalesJanJune  
    {  
        get  
        {  
            return salesJanJune;  
        }  
        set  
        {  
            salesJanJune = value;  
        }  
    }  
    public int SalesJulyDec  
    {  
        get  
        {  
            return salesJulyDec;  
        }  
        set  
        {  
            salesJulyDec = value;  
        }  
    }  
    public int Change  
    {  
        get  
        {  
            return change;  
        }  
        set  
        {  
            change = value;  
        }  
    }  
}
```


ASP.NET CORE

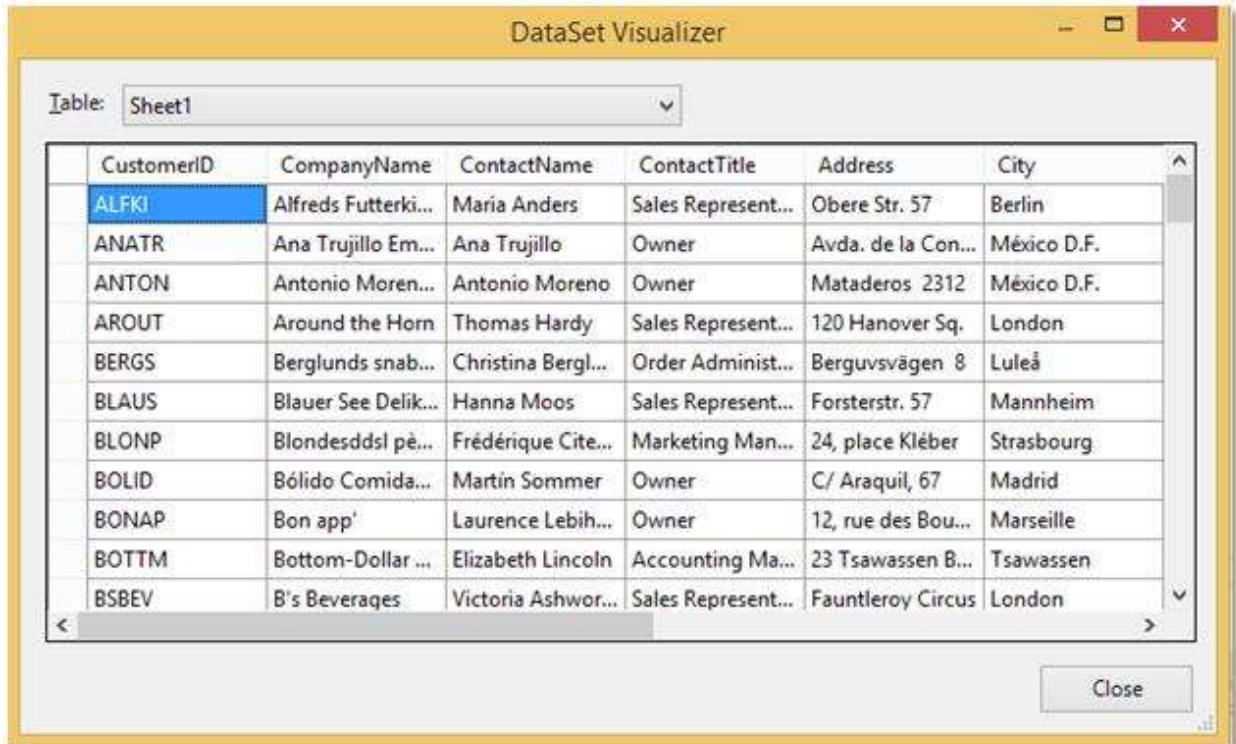
```
//XlsIO supports exporting of data from worksheet to data table from .NET
Standard 2.0
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("WorkbookWithData.xlsx",
    FileMode.Open, FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Export data from worksheet used range to a DataTable
    DataTable customersTable = worksheet.ExportDataTable(worksheet.UsedRange,
    ExcelExportDataTableOptions.ColumnNames |
    ExcelExportDataTableOptions.DetectColumnTypes);
    //Saving the workbook as stream
    FileStream file = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(file);
    file.Dispose();
}
```

XAMARIN

```
//XlsIO supports exporting of data from worksheet to data table from .NET
Standard 2.0 along with Windows Forms, WPF, ASP.NET and ASP.NET MVC
platforms alone.
//Exporting data from worksheet can be achieved using List as illustrated
below.
//To know more about exporting data from worksheet to various collection
objects, please refer xlsio/working-with-data section.
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream fileStream = assembly.GetManifestResourceStream("ExportSales.xlsx");
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet sheet = workbook.Worksheets[0];
    List<Sales> data = sheet.ExportData<Sales>(1, 1, 41, 4);
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Create
        Sheet.xlsx", "application/msexcel", stream);
    }
    else
```

```
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("CreateSheet.xlsx",
"application/msexcel", stream);
}
}
//Sales details
public class Sales
{
private string salesPerson;
private int salesJanJune;
private int salesJulyDec;
private int change;
public string SalesPerson
{
get
{
return salesPerson;
}
set
{
salesPerson = value;
}
}
public int SalesJanJune
{
get
{
return salesJanJune;
}
set
{
salesJanJune = value;
}
}
public int SalesJulyDec
{
get
{
return salesJulyDec;
}
set
{
salesJulyDec = value;
}
}
public int Change
{
get
{
return change;
}
set
{
change = value;
}
}
}
```

The following screenshot shows the DataTable of previous code.



DataSet Visualizer

Table: Sheet1

CustomerID	CompanyName	ContactName	ContactTitle	Address	City
ALFKI	Alfreds Futterki...	Maria Anders	Sales Represent...	Obere Str. 57	Berlin
ANATR	Ana Trujillo Em...	Ana Trujillo	Owner	Avda. de la Con...	México D.F.
ANTON	Antonio Moren...	Antonio Moreno	Owner	Mataderos 2312	México D.F.
AROUT	Around the Horn	Thomas Hardy	Sales Represent...	120 Hanover Sq.	London
BERGS	Berglunds snab...	Christina Bergl...	Order Administ...	Berguvsvägen 8	Luleå
BLAUS	Blauer See Delik...	Hanna Moos	Sales Represent...	Forsterstr. 57	Mannheim
BLONP	Blondesddsl pè...	Frédérique Cite...	Marketing Man...	24, place Kléber	Strasbourg
BOLID	Bólido Comida...	Martin Sommer	Owner	C/ Araquil, 67	Madrid
BONAP	Bon app'	Laurence Lebih...	Owner	12, rue des Bou...	Marseille
BOTTM	Bottom-Dollar ...	Elizabeth Lincoln	Accounting Ma...	23 Tsawassen B...	Tsawassen
BSBEV	B's Beverages	Victoria Ashwor...	Sales Represent...	Fauntleroy Circus	London

Close

You can refer various exporting options in the “Exporting from Worksheet to Data Table” section.

Template based data filling using Template Markers

A template marker is a special marker symbol that allows to generate a document by filling data in an Excel template from data source. This marker automatically maps the column name in the data source and names of the marker fields in the Excel template document and fills the data (text or image).

This functionality supports the following data sources.

- Collection Objects
- DataTable
- Array

Each marker starts with a prefix "%", which is followed by a **MarkerVariable** and its **property**. The arguments are delimited by semicolon (;). The following syntax shows the usage of marker in input template document.

```
%<MarkerVariable>.<Property>
```

For example: %Reports.SalesPerson

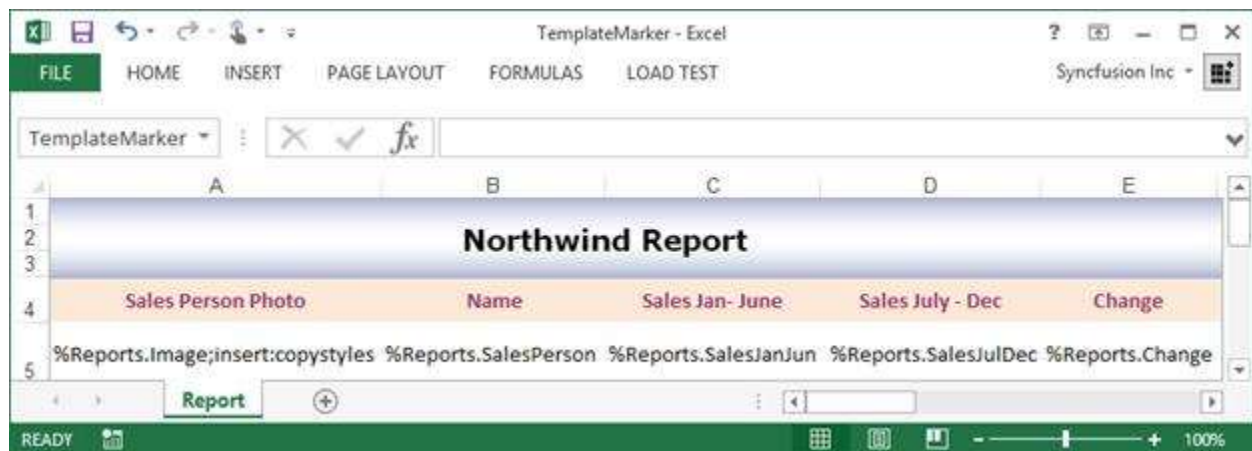
Use the following syntax to maintain row formats while filling data.

%<MarkerVariable>.<Property>;insert:copystyles

For example: %Reports.SalesPerson;insert:copystyles

Find more details in [Template marker section for arguments](#)

For example – let’s consider that you have a template document as shown below.



[Download input template](#)

The following code snippet shows how to use template markers with objects.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("TemplateMarker.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create template marker processor for the workbook
    ITemplateMarkersProcessor marker =
        workbook.CreateTemplateMarkersProcessor();
    //GetSalesReports method returns list of sales persons and their reports
    IList<Report> reports = GetSalesReports();
    //Adding reports collection to marker variables
    //Where the name should match with the input template
    marker.AddVariable("Reports", reports);
    //Applying Markers
    marker.ApplyMarkers();
    //Saving the workbook
    workbook.SaveAs("TemplateMarkerResult.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
```

```

Dim workbook As IWorkbook =
application.Workbooks.Open("TemplateMarker.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Create template marker processor for the workbook
Dim marker As ITemplateMarkersProcessor =
workbook.CreateTemplateMarkersProcessor()
'GetSalesReports method returns list of sales persons and their reports
Dim reports As IList(Of Report) = GetSalesReports()
'Adding reports collection to marker variables
'Where the name should match with the input template
marker.AddVariable("Reports", reports)
'Applying Markers
marker.ApplyMarkers()
'Saving the workbook
workbook.SaveAs("TemplateMarkerResult.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile openFile = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(openFile);
IWorksheet worksheet = workbook.Worksheets[0];
//Create template marker processor for the workbook
ITemplateMarkersProcessor marker =
workbook.CreateTemplateMarkersProcessor();
//GetSalesReports method returns list of sales persons and their reports
IList<Report> reports = GetSalesReports();
//Adding reports collection to marker variables
//Where the name should match with the input template
marker.AddVariable("Reports", reports);
//Applying Markers
marker.ApplyMarkers();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "TemplateMarkerResult";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("TemplateMarker.xlsx",
        FileMode.Open, FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    //Create template marker processor for the workbook
    ITemplateMarkersProcessor marker =
        workbook.CreateTemplateMarkersProcessor();
    //GetSalesReports method returns list of sales persons and their reports
    IList<Report> reports = GetSalesReports();
    //Adding reports collection to marker variables
    //Where the name should match with the input template
    marker.AddVariable("Reports", reports);
    //Applying Markers
    marker.ApplyMarkers();
    //Saving the workbook as stream
    FileStream file = new FileStream("TemplateMarkerResult.xlsx",
        FileMode.Create, FileAccess.ReadWrite);
    workbook.SaveAs(file);
    file.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.TemplateMarker.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    //Create template marker processor for the workbook
    ITemplateMarkersProcessor marker =
        workbook.CreateTemplateMarkersProcessor();
    //GetSalesReports method returns list of sales persons and their reports
    IList<Report> reports = GetSalesReports();
    //Adding reports collection to marker variables
    //Where the name should match with the input template
    marker.AddVariable("Reports", reports);
    //Applying Markers
    marker.ApplyMarkers();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {

```

```

Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("TemplateMarkerResult.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TemplateMarkerResult.xlsx", "application/msexcel", stream);
}
}

```

The following code snippet provides supporting methods and classes for the previous code.

C#

```

//Gets a list of sales reports
private static List<Report> GetSalesReports()
{
List<Report> reports = new List<Report>();
reports.Add(new Report("Andy Bernard", "45000", "58000", 29, "Andy.jpg"));
reports.Add(new Report("Jim Halpert", "34000", "65000", 91, "Jim.png"));
reports.Add(new Report("Karen Fillippelli", "75000", "64000", -14,
"Karen.jpg"));
reports.Add(new Report("Phyllis Lapin", "56500", "33600", -40,
"Phyllis.png"));
reports.Add(new Report("Stanley Hudson", "46500", "52000", 12,
"Stanley.jpg"));
return reports;
}
//Sales report
public class Report
{
public string SalesPerson { get; set; }
public string SalesJanJun { get; set; }
public string SalesJulDec { get; set; }
public int Change { get; set; }
public byte[] Image { get; set; }
public Report(string name, string janToJun, string julToDec, int change,
string imagePath)
{
SalesPerson = name;
SalesJanJun = janToJun;
SalesJulDec = julToDec;
Change = change;
Image = File.ReadAllBytes(imagePath);
}
}

```

VB.NET

```

'Gets a list of sales reports
Private Function GetSalesReports() As List(Of Report)
Dim reports As New List(Of Report)()
reports.Add(New Report("Andy Bernard", "45000", "58000", 29, "Andy.jpg"))
reports.Add(New Report("Jim Halpert", "34000", "65000", 91, "Jim.png"))
reports.Add(New Report("Karen Fillippelli", "75000", "64000", -14,
"Karen.jpg"))

```

```

reports.Add(New Report("Phyllis Lapin", "56500", "33600", -40,
"Phyllis.png"))
reports.Add(New Report("Stanley Hudson", "46500", "52000", 12,
"Stanley.jpg"))
Return reports
End Function
'Sales report
Public Class Report
Public Property SalesPerson() As String
Get
Return m_SalesPerson
End Get
Set(value As String)
m_SalesPerson = Value
End Set
End Property
Private m_SalesPerson As String
Public Property SalesJanJun() As String
Get
Return m_SalesJanJun
End Get
Set(value As String)
m_SalesJanJun = Value
End Set
End Property
Private m_SalesJanJun As String
Public Property SalesJulDec() As String
Get
Return m_SalesJulDec
End Get
Set(value As String)
m_SalesJulDec = Value
End Set
End Property
Private m_SalesJulDec As String
Public Property Change() As Integer
Get
Return m_Change
End Get
Set(value As Integer)
m_Change = Value
End Set
End Property
Private m_Change As Integer
Public Property Image() As Byte()
Get
Return m_Image
End Get
Set(value As Byte())
m_Image = Value
End Set
End Property
Private m_Image As Byte()
Public Sub New(name As String, janToJun As String, julToDec As String,
change As Integer, imagePath As String)
SalesPerson = name
SalesJanJun = janToJun

```



```

SalesJulDec = julToDec
change = change
Image = File.ReadAllBytes(imagePath)
End Sub
End Class

```

UWP

```

//Gets a list of sales reports
private static List<Report> GetSalesReports()
{
    List<Report> reports = new List<Report>();
    reports.Add(new Report("Andy Bernard", "45000", "58000", 29, "Andy.jpg"));
    reports.Add(new Report("Jim Halpert", "34000", "65000", 91, "Jim.png"));
    reports.Add(new Report("Karen Fillippelli", "75000", "64000", -14,
        "Karen.jpg"));
    reports.Add(new Report("Phyllis Lapin", "56500", "33600", -40,
        "Phyllis.png"));
    reports.Add(new Report("Stanley Hudson", "46500", "52000", 12,
        "Stanley.jpg"));
    return reports;
}

//Sales report
public class Report
{
    public string SalesPerson { get; set; }
    public string SalesJanJun { get; set; }
    public string SalesJulDec { get; set; }
    public int Change { get; set; }
    public byte[] Image { get; set; }
    public Report(string name, string janToJun, string julToDec, int change,
        string imagePath)
    {
        SalesPerson = name;
        SalesJanJun = janToJun;
        SalesJulDec = julToDec;
        Change = change;
        Image = GetImage(imagePath);
    }
    private byte[] GetImage(string imagePath)
    {
        Assembly assembly = typeof(App).GetTypeInfo().Assembly;
        Stream imageStream = assembly.GetManifestResourceStream("UWP.Data." +
            imagePath);
        using (BinaryReader br = new BinaryReader(imageStream))
        {
            return br.ReadBytes((int)imageStream.Length);
        }
    }
}

```

ASP.NET CORE

```

//Gets a list of sales reports
private static List<Report> GetSalesReports()
{

```

```

List<Report> reports = new List<Report>();
reports.Add(new Report("Andy Bernard", "45000", "58000", 29, "Andy.jpg"));
reports.Add(new Report("Jim Halpert", "34000", "65000", 91, "Jim.png"));
reports.Add(new Report("Karen Fillippelli", "75000", "64000", -14,
"Karen.jpg"));
reports.Add(new Report("Phyllis Lapin", "56500", "33600", -40,
"Phyllis.png"));
reports.Add(new Report("Stanley Hudson", "46500", "52000", 12,
"Stanley.jpg"));
return reports;
}
//Sales report
public class Report
{
public string SalesPerson { get; set; }
public string SalesJanJun { get; set; }
public string SalesJulDec { get; set; }
public int Change { get; set; }
public byte[] Image { get; set; }
public Report(string name, string janToJun, string julToDec, int change,
string imagePath)
{
SalesPerson = name;
SalesJanJun = janToJun;
SalesJulDec = julToDec;
Change = change;
Image = File.ReadAllBytes(imagePath);
}
}

```

XAMARIN

```

//Gets a list of sales reports
private static List<Report> GetSalesReports()
{
List<Report> reports = new List<Report>();
reports.Add(new Report("Andy Bernard", "45000", "58000", 29, "Andy.jpg"));
reports.Add(new Report("Jim Halpert", "34000", "65000", 91, "Jim.png"));
reports.Add(new Report("Karen Fillippelli", "75000", "64000", -14,
"Karen.jpg"));
reports.Add(new Report("Phyllis Lapin", "56500", "33600", -40,
"Phyllis.png"));
reports.Add(new Report("Stanley Hudson", "46500", "52000", 12,
"Stanley.jpg"));
return reports;
}
//Sales report
public class Report
{
public string SalesPerson { get; set; }
public string SalesJanJun { get; set; }
public string SalesJulDec { get; set; }
public int Change { get; set; }
public byte[] Image { get; set; }
public Report(string name, string janToJun, string julToDec, int change,
string imagePath)

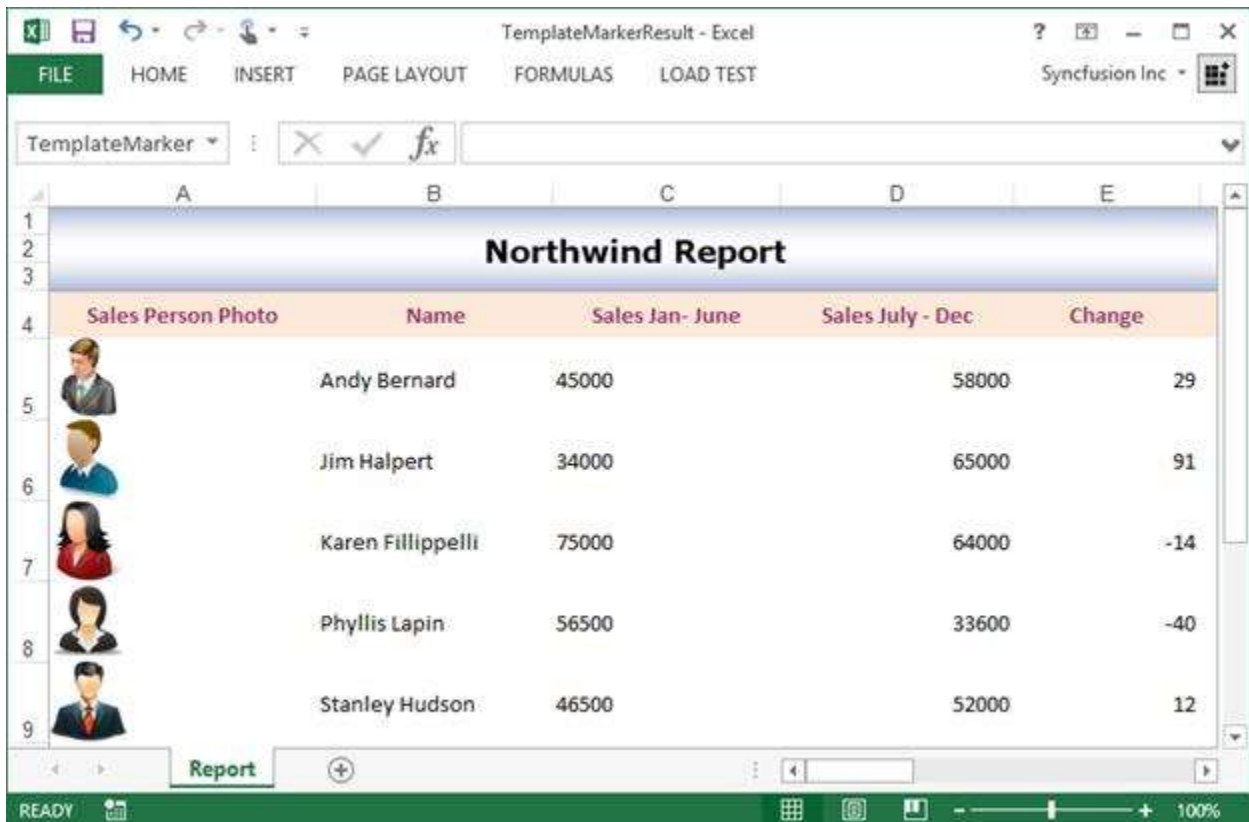
```






```

{
    SalesPerson = name;
    SalesJanJun = janToJun;
    SalesJulDec = julToDec;
    Change = change;
    Image = GetImage(imagePath);
}
private byte[] GetImage(string imagePath)
{
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream imageStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template." +
        imagePath);
    using (BinaryReader reader = new BinaryReader(imageStream))
    {
        return reader.ReadBytes((int)imageStream.Length);
    }
}
}
}

```

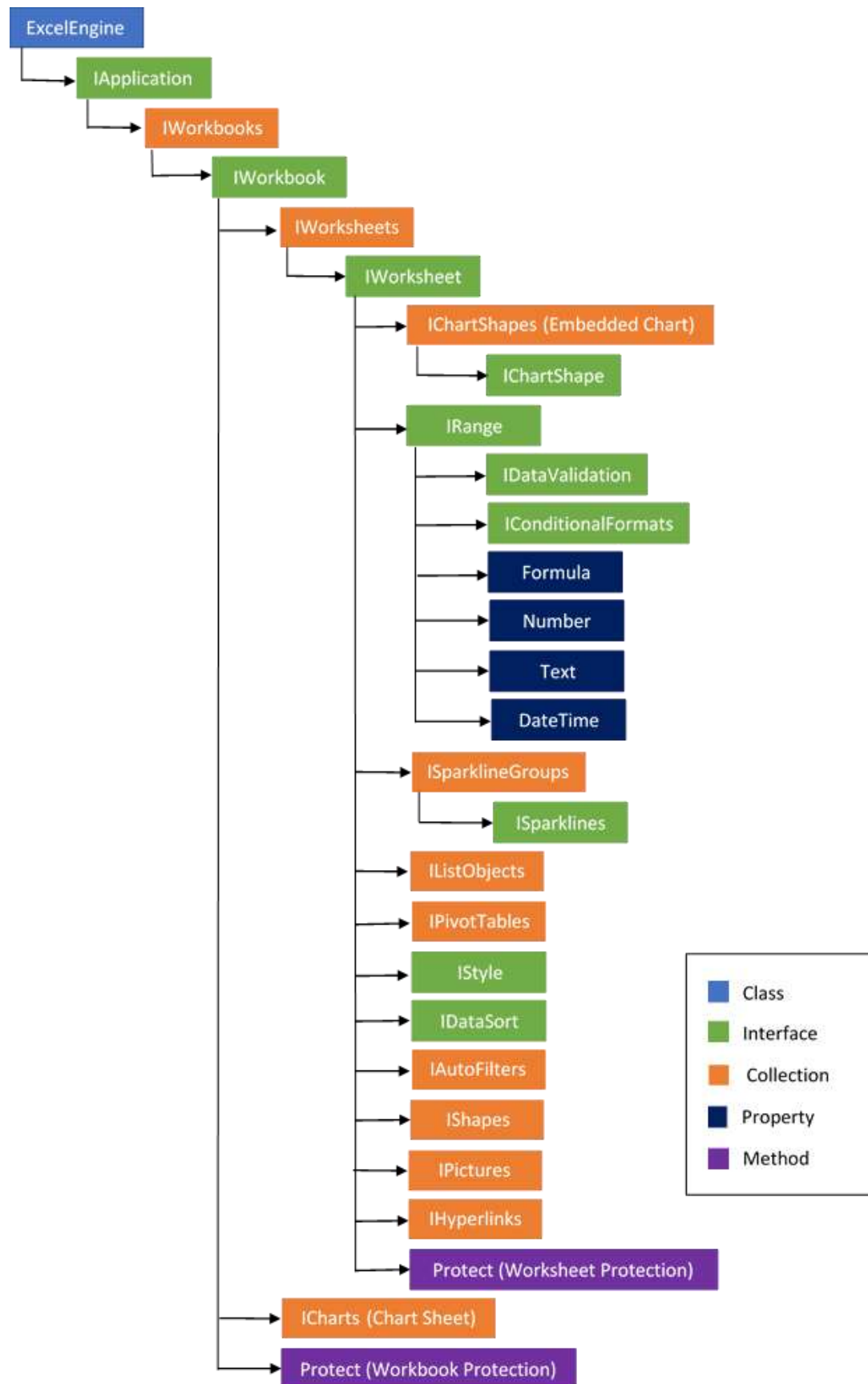
The resultant document looks as follows:



Northwind Report				
Sales Person Photo	Name	Sales Jan- June	Sales July - Dec	Change
	Andy Bernard	45000	58000	29
	Jim Halpert	34000	65000	91
	Karen Fillippelli	75000	64000	-14
	Phyllis Lapin	56500	33600	-40
	Stanley Hudson	46500	52000	12

Document Object Model

When an existing document is opened or a new document is created, the XlsIO library creates a **Document Object Model** (DOM) of the document in main memory. This object model can be used to manipulate the document as needed.



Object Reference Links

Object	Reference Link
ExcelEngine	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.ExcelEngine.html
IApplication	https://help.syncfusion.com/cr/cref_files/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IApplication.html
IWorkbooks	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IWorkbooks.html
IWorkbook	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IWorkbook.html
IWorksheets	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IWorksheets.html
IWorksheet	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IWorksheet.html
ICChartShapes (Embedded Chart)	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.ICChartShapes.html
ICChartShape	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.ICChartShape.html
IRange	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IRange.html
IDataValidation	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IDataValidation.html
ICConditionalFormats	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.ICConditionalFormats.html
Formula	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IRange~Formula.html
Number	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IRange~Number.html
Text	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IRange~Text.html

DateTime	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IRange~DateTime.html
ISparklineGroups	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.ISparklineGroups.html
ISparklines	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.ISparklines.html
IListObjects	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IListObjects.html
IPivotTables	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IPivotTables.html
IStyle	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IStyle.html
IDataSort	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IDataSort.html
IAutoFilters	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IAutoFilters.html
IShapes	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IShapes.html
IPictures	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IPictures.html
IHyperlinks	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IHyperLinks.html
Protect (Worksheet Protection)	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.ITabSheet~Protect.html
ICharts (Chart Sheet)	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.ICharts.html
Protect (Workbook Protection)	https://help.syncfusion.com/cr/file-formats/Syncfusion.XlsIO.Base~Syncfusion.XlsIO.IWorkbook~Protect.html

Loading and Saving Workbook

Opening an existing workbook

You can open an existing workbook by using the Open method of IWorkbooks interface.

C#

```
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//Loads or open an existing workbook through Open method of IWorkbooks
IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputFileName);
```

VB.NET

```
'Creates a new instance for ExcelEngine
Dim excelEngine As New ExcelEngine()
'Loads or open an existing workbook through Open method of IWorkbooks
Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Open(inputFileName)
```

UWP

```
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an existing workbook
IWorkbook workbook = await
excelEngine.Excel.Workbooks.OpenAsync(inputStorageFile);
```

ASP.NET CORE

```
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//Loads or open an existing workbook through Open method of IWorkbooks
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputStream);
```

XAMARIN

```
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(inputFilePath);
//Loads or open an existing workbook through Open method of IWorkbooks
IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputStream);
```

Tips: Files parsing can be optimized by setting **IApplication.UseFastRecordParsing = false** or **true** (true – fast mode, but less error checks and false – slower but more reliable).

C#

```
IApplication application = excelEngine.Excel;
//Optimize parsing
```

```
application.UseFastRecordParsing = true;
```

VB.NET

```
Dim application As IApplication = excelEngine.Excel
'Optimize parsing
application.UseFastRecordParsing = True
```

UWP

```
IApplication application = excelEngine.Excel;
//Optimize parsing
application.UseFastRecordParsing = true;
```

ASP.NET CORE

```
IApplication application = excelEngine.Excel;
//Optimize parsing
application.UseFastRecordParsing = true;
```

XAMARIN

```
IApplication application = excelEngine.Excel;
//Optimize parsing
application.UseFastRecordParsing = true;
```

Opening an existing workbook from Stream

You can open an existing workbook from stream by using the overloads of Open methods of IWorkbooks interface.

C#

```
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//Load the file into stream
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
//Loads or open an existing workbook through Open method of IWorkbooks
IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputStream);
```

VB.NET

```
'Creates a new instance for ExcelEngine
Dim excelEngine As New ExcelEngine()
'Load the file into stream
Dim inputStream As FileStream = New FileStream(inputFileName, FileMode.Open)
'Loads or open an existing workbook through Open method of IWorkbooks
Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Open(inputStream)
```

UWP

```
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//Load the file into stream
```



```
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
//Loads or open an existing workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream);
```

ASP.NET CORE

```
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//Load the file into stream
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
//Loads or open an existing workbook through Open method of IWorkbooks
IWorkbook workbook = application.Workbooks.Open(inputStream);
```

XAMARIN

```
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(inputFilePath);
//Loads or open an existing workbook through Open method of IWorkbooks
IWorkbook workbook = application.Workbooks.Open(inputStream);
```

Saving a Excel workbook to file system

You can save the created or manipulated workbook to file system using Save method of IWorkbook interface. The workbook is saved in the XLS/XLSX format based on the application/workbook version specified, whereas saved in Excel 97-2003 (*.xls) format by default.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Loads or open an existing workbook through Open method of IWorkbooks
    IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputFileName);
    //To-Do some manipulation
    //To-Do some manipulation
    //Set the version of the workbook
    workbook.Version = ExcelVersion.Excel2013;
    //Save the workbook in file system as XLSX format
    workbook.SaveAs(outputFileName);
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
'Loads or open an existing workbook through Open method of IWorkbooks
Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Open(inputFileName)
'To-Do some manipulation
'To-Do some manipulation
'Set the version of the workbook
workbook.Version = ExcelVersion.Excel2013
'Save the workbook in file system as XLSX format
workbook.SaveAs(outputFileName)
End Using
```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    //Creates a storage file from FileOpenPicker
    StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
    //Loads or open an existing workbook
    IWorkbook workbook = await
    excelEngine.Excel.Workbooks.OpenAsync(inputStorageFile);
    //To-Do some manipulation
    //To-Do some manipulation
    //Set the version of the workbook
    workbook.Version = ExcelVersion.Excel2013;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = OutputFileName;
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile outputStorageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(outputStorageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Loads or open an existing workbook
    FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
    IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputStream);
    //To-Do some manipulation
    //To-Do some manipulation
    //Set the version of the workbook
    workbook.Version = ExcelVersion.Excel2013;
    //Saving the workbook
    FileStream outputStream = new FileStream(outputFileName, FileMode.Create);
    workbook.SaveAs(outputStream);
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream = assembly.GetManifestResourceStream(inputFilePath);
    //Loads or open an existing workbook through Open method of IWorkbooks

```

```

IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputStream);
//To-Do some manipulation
//To-Do some manipulation
//Set the version of the workbook
workbook.Version = ExcelVersion.Excel2013;
//Saving the workbook
MemoryStream outputStream = new MemoryStream();
workbook.SaveAs(outputStream);
outputStream.Position = 0;
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(outputF
ileName, "application/msexcel", outputStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(outputFileName,
"application/msexcel", outputStream);
}
}

```

Saving an Excel workbook to stream

You can also save the created or manipulated workbook to stream using overloads of Save methods

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
//Loads or open an existing workbook through Open method of IWorkbooks
IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputFileName);
//To-Do some manipulation
//To-Do some manipulation
//Set the version of the workbook
workbook.Version = ExcelVersion.Excel2013;
//Save the workbook as stream
MemoryStream outputStream = new MemoryStream();
workbook.SaveAs(outputStream);
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
'Loads or open an existing workbook through Open method of IWorkbooks
Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Open(inputFileName)
'To-Do some manipulation
'To-Do some manipulation
'Set the version of the workbook
workbook.Version = ExcelVersion.Excel2013
'Save the workbook as stream
Dim outputStream As MemoryStream = New MemoryStream
workbook.SaveAs(outputStream)
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    //Creates a storage file from FileOpenPicker
    StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
    //Loads or open an existing workbook
    IWorkbook workbook = await
    excelEngine.Excel.Workbooks.OpenAsync(inputStorageFile);
    //To-Do some manipulation
    //To-Do some manipulation
    //Set the version of the workbook
    workbook.Version = ExcelVersion.Excel2013;
    //Saves changes to the specified storage file
    MemoryStream outputStream = new MemoryStream();
    await workbook.SaveAsAsync(outputStream);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Loads or open an existing workbook through Open method of IWorkbooks
    FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
    IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputStream);
    //To-Do some manipulation
    //To-Do some manipulation
    //Set the version of the workbook
    workbook.Version = ExcelVersion.Excel2013;
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream = assembly.GetManifestResourceStream(inputFilePath);
    //Loads or open an existing workbook through Open method of IWorkbooks
    IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputStream);
    //To-Do some manipulation
    //To-Do some manipulation
    //Set the version of the workbook
    workbook.Version = ExcelVersion.Excel2013;
    //Save the workbook as stream
    MemoryStream outputStream = new MemoryStream();
}

```

```
workbook.SaveAs(outputStream);
}
```

Sending to a client browser

You can save & send the workbook to a client browser from a web site or web application by invoking the below shown overload of Save method. This method explicitly make use of an instance of [HttpResponse](#) as its parameter in order to stream the workbook to client browser. So this overload is suitable for web application which references System.Web assembly.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Loads or open an existing workbook through Open method of IWorkbooks
    IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputFileName);
    //To-Do some manipulation
    //To-Do some manipulation
    //Set the version of the workbook
    workbook.Version = ExcelVersion.Excel2013;
    //Save the workbook in file system
    workbook.SaveAs(OutputFileName, Response, ExcelDownloadType.Open);
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
    'Loads or open an existing workbook through Open method of IWorkbooks
    Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Open(inputFileName)
    'To-Do some manipulation
    'To-Do some manipulation
    'Set the version of the workbook
    workbook.Version = ExcelVersion.Excel2013
    'Save the workbook in file system
    workbook.SaveAs(OutputFileName, Response, ExcelDownloadType.Open)
End Using
```

UWP

```
//Saving and sending the workbook to a client browser from a web site is
suitable for web applications alone.
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Loads or open an existing workbook through Open method of IWorkbooks
    FileStream inputStream = new FileStream(inputFilePath, FileMode.Open);
    IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputStream);
    //To-Do some manipulation
    //To-Do some manipulation
    //Initialize content type
    string ContentType = null;
    //Set the version of the workbook
    workbook.Version = ExcelVersion.Excel2013;
}
```

```

ContentType = "Application/msexcel";
//Save the workbook to stream
MemoryStream outputStream = new MemoryStream();
workbook.SaveAs(outputStream);
outputStream.Position = 0;
//Return the file with content type
return File(outputStream, ContentType, outputFileName);
}

```

XAMARIN

```

//Saving and sending the workbook to a client browser from a web site is
suitable for web applications alone.

```

Closing a workbook

Once after the workbook manipulation and save operation are completed, you should close the instance of IWorkbook and dispose the instance of ExcelEngine, in order to release all the memory consumed by XlsIO's DOM. The following code snippet illustrates how to close the instance of IWorkbook and dispose the instance of ExcelEngine.

Note: If the new instance for ExcelEngine is created in using statement, then there is no need to closing workbook and disposing excelEngine.

C#

```

//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//Loads or open an existing workbook through Open method of IWorkbooks
IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputFileName);
//To-Do some manipulation
//To-Do some manipulation
//Set the version of the workbook
workbook.Version = ExcelVersion.Excel2013;
//Save the workbook in file system
workbook.SaveAs(outputFileName);
//Close the instance of IWorkbook
workbook.Close();
//Dispose the instance of ExcelEngine
excelEngine.Dispose();

```

VB.NET

```

'Creates a new instance for ExcelEngine
Dim excelEngine As New ExcelEngine()
'Loads or open an existing workbook through Open method of IWorkbooks
Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Open(inputFileName)
'To-Do some manipulation
'To-Do some manipulation
'Set the version of the workbook
workbook.Version = ExcelVersion.Excel2013
'Save the workbook in file system
workbook.SaveAs(outputFileName)
'Close the instance of IWorkbook
workbook.Close()
'Dispose the instance of ExcelEngine

```

```
excelEngine.Dispose();
```

UWP

```
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
//Creates a storage file from FileOpenPicker
StorageFile inputStorageFile = await openPicker.PickSingleFileAsync();
//Loads or open an existing workbook
IWorkbook workbook =
excelEngine.Excel.Workbooks.OpenAsync(inputStorageFile);
//To-Do some manipulation
//To-Do some manipulation
//Set the version of the workbook
workbook.Version = ExcelVersion.Excel2013;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(outputStorageFile);
//Close the instance of IWorkbook
workbook.Close();
//Dispose the instance of ExcelEngine
excelEngine.Dispose();
```

ASP.NET CORE

```
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//Loads or open an existing workbook through Open method of IWorkbooks
FileStream inputStream = new FileStream(inputFileName, FileMode.Open);
IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputStream);
//To-Do some manipulation
//To-Do some manipulation
//Set the version of the workbook
workbook.Version = ExcelVersion.Excel2013;
//Save the workbook as stream
FileStream outputStream = new FileStream("Output.xlsx", FileMode.Create);
workbook.SaveAs(outputStream);
//Close the instance of IWorkbook
workbook.Close();
//Dispose the instance of ExcelEngine
excelEngine.Dispose();
```

XAMARIN

```

//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream(inputFilePath);
//Loads or open an existing workbook through Open method of IWorkbooks
IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputStream);
//To-Do some manipulation
//To-Do some manipulation
//Set the version of the workbook
workbook.Version = ExcelVersion.Excel2013;
//Save the workbook as stream
MemoryStream outputStream = new MemoryStream();
workbook.SaveAs(outputStream);
outputStream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", outputStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", outputStream);
}
//Close the instance of IWorkbook
workbook.Close();
//Dispose the instance of ExcelEngine
excelEngine.Dispose();

```

Tips: You can use `ThrowNotSavedOnDestroy` property of `ExcelEngine` object to prevent the data loss while unfortunately closing the workbook or disposing excel engine without saving contents. If it is set to true, then `ExcelWorkbookNotSavedException` will be thrown when you forgot to save the workbook before closing them. Following code illustrates how to set `ThrowNotSavedOnDestroy` property of `ExcelEngine` object.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
//No exception will be thrown if there are unsaved workbooks
excelEngine.ThrowNotSavedOnDestroy = true;

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
'No exception will be thrown if there are unsaved workbooks
excelEngine.ThrowNotSavedOnDestroy = True

```

UWP


```
ExcelEngine excelEngine = new ExcelEngine();
//No exception will be thrown if there are unsaved workbooks
excelEngine.ThrowNotSavedOnDestroy = true;
```

ASP.NET CORE

```
ExcelEngine excelEngine = new ExcelEngine();
//No exception will be thrown if there are unsaved workbooks
excelEngine.ThrowNotSavedOnDestroy = true;
```

XAMARIN

```
ExcelEngine excelEngine = new ExcelEngine();
//No exception will be thrown if there are unsaved workbooks
excelEngine.ThrowNotSavedOnDestroy = true;
```

Working with Excel Worksheet

A Workbook contains a collection of worksheets where the actual contents resides and IWorksheet instance represents a worksheet. With XlsIO, You can add and manipulate worksheets.

Create a Worksheet

You can add a new worksheet into the Workbook through Create method of IWorkbook interface. You can also specify the required number of worksheets, if not specified, XlsIO will create three worksheets by default.

The following code snippet shows how to create worksheets within a workbook.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //The new workbook will have 5 worksheets
    IWorkbook workbook = application.Workbooks.Create(5);
    //Creating a Sheet
    IWorksheet sheet = workbook.Worksheets.Create();
    //Creating a Sheet with name "Sample"
    IWorksheet namedSheet = workbook.Worksheets.Create("Sample");
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
'The new workbook will have 5 worksheets
Dim workbook As IWorkbook = application.Workbooks.Create(5)
'Creating a sheet
Dim sheet As IWorksheet = workbook.Worksheets.Create()
'Creating a Sheet with name "Sample"
Dim namedSheet As IWorksheet = workbook.Worksheets.Create("Sample")
workbook.SaveAs("Output.xlsx")
```

End Using**UWP**

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //The new workbook will have 5 worksheets.
    IWorkbook workbook = application.Workbooks.Create(5);
    //Creating a Sheet
    IWorksheet sheet = workbook.Worksheets.Create();
    //Creating a Sheet with name "Sample"
    IWorksheet namedSheet = workbook.Worksheets.Create("Sample");
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //The new workbook will have 5 worksheets
    IWorkbook workbook = application.Workbooks.Create(5);
    //Creating a Sheet
    IWorksheet sheet = workbook.Worksheets.Create();
    //Creating a Sheet with name "Sample"
    IWorksheet namedSheet = workbook.Worksheets.Create("Sample");
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //The new workbook will have 5 worksheets.
    IWorkbook workbook = application.Workbooks.Create(5);
    //Creating a Sheet.
    IWorksheet sheet = workbook.Worksheets.Create();
}

```

```

//Creating a Sheet with name "Sample"
IWorksheet namedSheet = workbook.Worksheets.Create("Sample");
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Access a Worksheet

Worksheets collection holds one or more worksheets present in a workbook. Accessing a particular worksheet can be done by the following ways.

1. Specifying the index
2. Specifying the sheet name.

The below codes illustrate how to access a worksheet from its worksheets collection.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(2);
//Accessing via index
IWorksheet sheet = workbook.Worksheets[0];
//Accessing via sheet Name
IWorksheet NamedSheet = workbook.Worksheets["Sample"];
workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(2)
'Accessing via index
Dim sheet As IWorksheet = workbook.Worksheets(0)

```

```
'Accessing via Sheet Name
Dim NamedSheet As IWorksheet = workbook.Worksheets("Sample")
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    //Accessing via index
    IWorksheet sheet = workbook.Worksheets[0];
    //Accessing via sheet Name
    IWorksheet NamedSheet = workbook.Worksheets["Sample"];
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    //Accessing via index
    IWorksheet sheet = workbook.Worksheets[0];
    //Accessing via sheet Name
    IWorksheet NamedSheet = workbook.Worksheets["Sample"];
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    //Accessing via index
    IWorksheet sheet = workbook.Worksheets[0];
```

```

//Accessing via sheet Name
IWorksheet NamedSheet = workbook.Worksheets["Sample"];
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Tips: If the workbook contains multiple worksheet, then the parsing of the workbook will consume time. You can use **ExcelParseOptions.ParseWorksheetsOnDemand** in **IWorkbooks.Open** method which parses the worksheet only when it is accessed. This option can be used in a scenario where workbook contains multiple worksheets but you are going to use few worksheets among them.

C#

```

IWorkbook workbook =
application.Workbooks.Open(fileName, ExcelParseOptions.ParseWorksheetsOnDemand);

```

VB.NET

```

Dim workbook As IWorkbook = application.Workbooks.Open(fileName,
ExcelParseOptions.ParseWorksheetsOnDemand)

```

UWP

```

IWorkbook workbook = await
application.Workbooks.OpenAsync(workbookStorageFile, ExcelOpenType.Automatic,
ExcelParseOptions.ParseWorksheetsOnDemand);

```

ASP.NET CORE

```

IWorkbook workbook =
application.Workbooks.Open(workbookStream, ExcelParseOptions.ParseWorksheetsOnDemand);

```

XAMARIN

```
IWorkbook workbook =
application.Workbooks.Open(workbookStream, ExcelParseOptions.ParseWorksheetsOnDemand);
```

Remove a Worksheet

Deletes the worksheets from the workbook collection by accessing the worksheet.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    //Removing the sheet
    workbook.Worksheets[0].Remove();
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(2)
'Removing the sheet
workbook.Worksheets(0).Remove()
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    //Removing the sheet
    workbook.Worksheets[0].Remove();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx" });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
```

```

IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(2);
//Removing the sheet
workbook.Worksheets[0].Remove();
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    //Removing the sheet
    workbook.Worksheets[0].Remove();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
        "application/msexcel", stream);
    }
}

```

Move or Copy a Worksheet

Essential XlsIO allows to move or create a copy of a worksheet, and insert that worksheet before or after an existing worksheet in the workbook.

Copying Worksheets

You can copy a worksheet to one another workbook or within the same workbook.

The following code example illustrates how to copy a sheet with its entire contents to another workbook.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{

```

```

IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook sourceWorkbook =
application.Workbooks.Open("SourceWorkbookTemplate.xlsx");
IWorkbook destinationWorkbook =
application.Workbooks.Open("DestinationWorkbookTemplate.xlsx");
//Copy first worksheet from the Source workbook to the destination workbook
destinationWorkbook.Worksheets.AddCopy(sourceWorkbook.Worksheets[0]);
destinationWorkbook.ActiveSheetIndex = 1;
destinationWorkbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim sourceWorkbook As IWorkbook =
application.Workbooks.Open("SourceWorkbookTemplate.xlsx")
Dim destinationWorkbook As IWorkbook =
application.Workbooks.Open("DestinationWorkbookTemplate.xlsx")
'Copy first worksheet from the Source workbook to the destination workbook
destinationWorkbook.Worksheets.AddCopy(sourceWorkbook.Worksheets(0))
destinationWorkbook.ActiveSheetIndex = 1
destinationWorkbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile sourceFile = await openPicker.PickSingleFileAsync();
StorageFile destinationFile = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook sourceWorkbook = await
application.Workbooks.OpenAsync(sourceFile);
IWorkbook destinationWorkbook = await
application.Workbooks.OpenAsync(destinationFile);
//Copy first worksheet from the Source workbook to the destination workbook
destinationWorkbook.Worksheets.AddCopy(sourceWorkbook.Worksheets[0]);
destinationWorkbook.ActiveSheetIndex = 1;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();

```



```
//Saves changes to the specified storage file
await destinationWorkbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream sourceStream = new FileStream("SourceWorkbookTemplate.xlsx",
    FileMode.Open, FileAccess.Read);
    IWorkbook sourceWorkbook = application.Workbooks.Open(sourceStream);
    FileStream destinationStream = new
    FileStream("DestinationWorkbookTemplate.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook destinationWorkbook =
    application.Workbooks.Open(destinationStream);
    //Copy first worksheet from the Source workbook to the destination workbook
    destinationWorkbook.Worksheets.AddCopy(sourceWorkbook.Worksheets[0]);
    destinationWorkbook.ActiveSheetIndex = 1;
    //Saving the workbook as stream
    FileStream copiedStream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    destinationWorkbook.SaveAs(copiedStream);
    copiedStream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    /"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream sourceStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sou
    rceWorkbookTemplate.xlsx");
    IWorkbook sourceWorkbook = application.Workbooks.Open(sourceStream);
    Stream destinationStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Des
    tinationWorkbookTemplate.xlsx");
    IWorkbook destinationWorkbook =
    application.Workbooks.Open(destinationStream);
    //Copy first worksheet from the Source workbook to the destination workbook
    destinationWorkbook.Worksheets.AddCopy(sourceWorkbook.Worksheets[0]);
    destinationWorkbook.ActiveSheetIndex = 1;
    //Saving the workbook as stream
    MemoryStream copiedStream = new MemoryStream();
    destinationWorkbook.SaveAs(copiedStream);
    copiedStream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
}
```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", copiedStream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", copiedStream);
}
}

```

You can specify copy options while copying a worksheet, which helps to achieve customized copying by ignore the certain formatting. For more information about copy options, please refer **ExcelWorksheetCopyFlags** .

Moving a Worksheet

XlsIO allows moving worksheets from one position to another by using the **Move** method. The following code example illustrates how a worksheet is moved.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(3);
    IWorksheet sheet = workbook.Worksheets[0];
    //Move the Sheet
    sheet.Move(1);
    workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(3)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Move the sheet
sheet.Move(1)
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(3);
    IWorksheet sheet = workbook.Worksheets[0];
    //Move the Sheet
}

```

```

sheet.Move(1);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(3);
    IWorksheet sheet = workbook.Worksheets[0];
    //Move the Sheet
    sheet.Move(1);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(3);
    IWorksheet sheet = workbook.Worksheets[0];
    //Move the Sheet
    sheet.Move(1);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {

```

```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}
```

Highlight Worksheet Tabs

You can highlight the worksheet tab of a particular sheet to denote its importance. You can set the tab color through the **TabColor** property, as given below.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Highlighting sheet tab
    sheet.TabColor = ExcelKnownColors.Red;
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Highlighting sheet tab
sheet.TabColor = ExcelKnownColors.Red
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Highlighting sheet tab
    sheet.TabColor = ExcelKnownColors.Red;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Highlighting sheet tab
    sheet.TabColor = ExcelKnownColors.Red;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Highlighting sheet tab
    sheet.TabColor = ExcelKnownColors.Red;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
        "application/msexcel", stream);
    }
}

```

Freeze Panes

You can [freeze](#) a portion of the sheet to keep it visible while you scroll through the rest of the sheet. The following code snippet shows how to freeze panes through the FreezePanes method of **IRange**.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Applying Freeze Pane to the sheet by specifying a cell
    sheet.Range["B2"].FreezePanes();
    workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Applying Freeze Pane to the sheet by specifying a cell
sheet.Range("B2").FreezePanes()
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Applying Freeze Pane to the sheet by specifying a cell
    sheet.Range["B2"].FreezePanes();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Applying Freeze Pane to the sheet by specifying a cell
    sheet.Range["B2"].FreezePanes();
}

```

```
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Applying Freeze Pane to the sheet by specifying a cell
    sheet.Range["B2"].FreezePanes();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
        "application/msexcel", stream);
    }
}
```

You can set first visible row and first visible column in non-frozen area, by setting the FirstVisibleRow and FirstVisibleColumn as shown below

Note: FirstVisibleColumn and FirstVisibleRow indexes are "zero-based".

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["B2"].FreezePanes();
    //Set first visible row in the bottom pane
    sheet.FirstVisibleRow = 2;
    //Set first visible column in the right pane
    sheet.FirstVisibleColumn = 2;
}
```

```
workbook.SaveAs ("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
sheet.Range("B2").FreezePanes()
'Set first visible row in the bottom pane
sheet.FirstVisibleRow = 2
'Set first visible column in the right pane
sheet.FirstVisibleColumn = 2
workbook.SaveAs ("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
sheet.Range["B2"].FreezePanes();
//Set first visible row in the bottom pane
sheet.FirstVisibleRow = 2;
//Set first visible column in the right pane
sheet.FirstVisibleColumn = 2;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
sheet.Range["B2"].FreezePanes();
//Set first visible row in the bottom pane
sheet.FirstVisibleRow = 2;
//Set first visible column in the right pane
```



```

sheet.FirstVisibleColumn = 2;
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["B2"].FreezePanes();
    //Set first visible row in the bottom pane
    sheet.FirstVisibleRow = 2;
    //Set first visible column in the right pane
    sheet.FirstVisibleColumn = 2;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
    }
}

```

Unfreeze Panes

You can unfreeze panes in an Excel worksheet using the [RemovePanes](#) method of **IWorksheet** interface. Refer to the following complete code snippets.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Unfreeze panes in the worksheet
}

```

```
worksheet.RemovePanes();
workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Unfreeze panes in the worksheet
worksheet.RemovePanes()
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the file picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Unfreeze panes in the worksheet
    worksheet.RemovePanes();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
}
```

```
//Unfreeze panes in the worksheet
worksheet.RemovePanes();
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Unfreeze panes in the worksheet
    worksheet.RemovePanes();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx", "application/msexcel", stream);
    }
}
```

Split Panes

You can divide the window into different [panes](#) that each scroll separately. The following code snippets illustrates how to split the window through the **HorizontalSplit** and **VerticalSplit** properties.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
```

```

IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//split panes
sheet.FirstVisibleColumn = 5;
sheet.FirstVisibleRow = 11;
sheet.VerticalSplit = 1100;
sheet.HorizontalSplit = 1000;
sheet.ActivePane = 1;
workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Split Panes
sheet.FirstVisibleColumn = 5
sheet.FirstVisibleRow = 11
sheet.VerticalSplit = 1100
sheet.HorizontalSplit = 1000
sheet.ActivePane = 1
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//split panes
sheet.FirstVisibleColumn = 5;
sheet.FirstVisibleRow = 11;
sheet.VerticalSplit = 1100;
sheet.HorizontalSplit = 1000;
sheet.ActivePane = 1;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //split panes
    sheet.FirstVisibleColumn = 5;
    sheet.FirstVisibleRow = 11;
    sheet.VerticalSplit = 1100;
    sheet.HorizontalSplit = 1000;
    sheet.ActivePane = 1;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //split panes
    sheet.FirstVisibleColumn = 5;
    sheet.FirstVisibleRow = 11;
    sheet.VerticalSplit = 1100;
    sheet.HorizontalSplit = 1000;
    sheet.ActivePane = 1;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
        "application/msexcel", stream);
    }
}

```

Page Setup Settings

You can select the size, orientation of the paper, margins, page breaks, scaling, paper size, header/footer settings and background settings. The following code snippet shows how to set the page setup. To more about, please refer **Page setup**.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "PageBreak";
    //Set Horizontal Page Breaks
    sheet.HPageBreaks.Add(sheet.Range["A5"]);
    //Set Vertical Page Breaks
    sheet.VPageBreaks.Add(sheet.Range["B5"]);
    //Set print title
    sheet.PageSetup.PrintTitleColumns = "$B:$E";
    sheet.PageSetup.PrintTitleRows = "$2:$5";
    //Set Page Orientation as Portrait or Landscape
    sheet.PageSetup.Orientation = ExcelPageOrientation.Landscape;
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
sheet.Range("A1:M20").Text = "PageBreak"
' Set Horizontal Page Breaks
sheet.HPageBreaks.Add(sheet.Range("A5"))
' Set Vertical Page Breaks
sheet.VPageBreaks.Add(sheet.Range("B5"))
' Set print titles
sheet.PageSetup.PrintTitleColumns = "$B:$E"
sheet.PageSetup.PrintTitleRows = "$2:$5"
' Set Page Orientation as Portrait or Landscape
sheet.PageSetup.Orientation = ExcelPageOrientation.Landscape
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "PageBreak";
}
```

```

//Set Horizontal Page Breaks
sheet.HPageBreaks.Add(sheet.Range["A5"]);
//Set Vertical Page Breaks
sheet.VPageBreaks.Add(sheet.Range["B5"]);
//Set print title
sheet.PageSetup.PrintTitleColumns = "$B:$E"
sheet.PageSetup.PrintTitleRows = "$2:$5"
//Set Page Orientation as Portrait or Landscape
sheet.PageSetup.Orientation = ExcelPageOrientation.Landscape;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "PageBreak";
    //Set Horizontal Page Breaks
    sheet.HPageBreaks.Add(sheet.Range["A5"]);
    //Set Vertical Page Breaks
    sheet.VPageBreaks.Add(sheet.Range["B5"]);
    //Set print title
    sheet.PageSetup.PrintTitleColumns = "$B:$E";
    sheet.PageSetup.PrintTitleRows = "$2:$5";
    //Set Page Orientation as Portrait or Landscape
    sheet.PageSetup.Orientation = ExcelPageOrientation.Landscape;
    //Saving the workbook as stream
    FileStream stream = new FileStream("output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "PageBreak";
    //Set Horizontal Page Breaks

```

```

sheet.HPageBreaks.Add(sheet.Range["A5"]);
//Set Vertical Page Breaks.
sheet.VPageBreaks.Add(sheet.Range["B5"]);
//Set print title
sheet.PageSetup.PrintTitleColumns = "$B:$E";
sheet.PageSetup.PrintTitleRows = "$2:$5";
//Set Page Orientation as Portrait or Landscape
sheet.PageSetup.Orientation = ExcelPageOrientation.Landscape;
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Show or Hide Worksheet

The following code snippet shows how to hide the sheets using **Visibility** property.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(2);
IWorksheet sheet = workbook.Worksheets[0];
sheet.Range["A1:M20"].Text = "visibility";
//Set visibility
sheet.Visibility = WorksheetVisibility.Hidden;
workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(2)
Dim sheet As IWorksheet = workbook.Worksheets(0)
sheet.Range("A1:M20").Text = "Visibility"
'Set visibility
sheet.Visibility = WorksheetVisibility.Hidden

```



```
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "visibility";
    //Set visibility
    sheet.Visibility = WorksheetVisibility.Hidden;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "visibility";
    //Set visibility
    sheet.Visibility = WorksheetVisibility.Hidden;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "visibility";
    //Set visibility
    sheet.Visibility = WorksheetVisibility.Hidden;
```

```
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}
```

Activate a Worksheet

You can set a worksheet as active sheet in the workbook by using the **Activate** method.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Activate";
    //Activate the sheet
    sheet.Activate();
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(2)
Dim sheet As IWorksheet = workbook.Worksheets(0)
sheet.Range("A1:M20").Text = "Activate"
'Activate the sheet
sheet.Activate()
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
```

```

IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(2);
IWorksheet sheet = workbook.Worksheets[0];
sheet.Range["A1:M20"].Text = "Activate";
//Activate the sheet
sheet.Activate();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(2);
IWorksheet sheet = workbook.Worksheets[0];
sheet.Range["A1:M20"].Text = "Activate";
//Activate the sheet
sheet.Activate();
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(2);
IWorksheet sheet = workbook.Worksheets[0];
sheet.Range["A1:M20"].Text = "Activate";
//Activate the sheet
sheet.Activate();
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
}

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Show or Hide Worksheet Tabs

The following code snippet shows how to hide the worksheet tab using **DisplayWorkbookTabs** property.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(3);
IWorksheet sheet = workbook.Worksheets[0];
sheet.Range["A1:M20"].Text = "Tabs";
//Hide the tab
workbook.DisplayWorkbookTabs = false;
//set the display tab
workbook.DisplayedTab = 2;
workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(3)
Dim sheet As IWorksheet = workbook.Worksheets(0)
sheet.Range("A1:M20").Text = "Tabs"
'Hide the tab
workbook.DisplayWorkbookTabs = False
'set the display tab
workbook.DisplayedTab = 2
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(3);
IWorksheet sheet = workbook.Worksheets[0];

```

```

sheet.Range["A1:M20"].Text = "Tabs";
//Hide the tab
workbook.DisplayWorkbookTabs = false;
//set the display tab
workbook.DisplayedTab = 2;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(3);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Tabs";
    //Hide the tab
    workbook.DisplayWorkbookTabs = false;
    //set the display tab
    workbook.DisplayedTab = 2;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(3);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Tabs";
    //Hide the tab
    workbook.DisplayWorkbookTabs = false;
    //set the display tab
    workbook.DisplayedTab = 2;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
}

```

```
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}
```

View Settings

Show or Hide Row and Column Headers

You can show/hide row and column headings by using the **IsRowColumnHeadersVisible** property of **IWorksheet**.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
sheet.Range["A1:M20"].Text = "RowColumnHeader";
sheet.IsRowColumnHeadersVisible = false;
workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
sheet.Range("A1:M20").Text = "RowColumnHeader"
sheet.IsRowColumnHeadersVisible = False
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
```

```

sheet.Range["A1:M20"].Text = "RowColumnHeader";
sheet.IsRowColumnHeadersVisible = false;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "RowColumnHeader";
    sheet.IsRowColumnHeadersVisible = false;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "RowColumnHeader";
    sheet.IsRowColumnHeadersVisible = false;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else

```

```
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}
```

Show or Hide Grid Lines

The following code snippet shows how to hide the grid lines using `IsGridLinesVisible` property.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Gridlines";
    //Hide grid line
    sheet.IsGridLinesVisible = false;
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
sheet.Range("A1:M20").Text = "GridLines"
'Hide grid line
sheet.IsGridLinesVisible = False
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Gridlines";
    //Hide grid line
    sheet.IsGridLinesVisible = false;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
}
```



```
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Gridlines";
    //Hide grid line
    sheet.IsGridLinesVisible = false;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Gridlines";
    //Hide grid line
    sheet.IsGridLinesVisible = false;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
        "application/msexcel", stream);
    }
}
```

Set Zoom Level

The following code snippet shows how to set the zoom level by using **Zoom** property.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Zoom level";
    //set zoom percentage
    sheet.Zoom = 70;
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
sheet.Range("A1:M20").Text = "Zoom level"
'set Zoom percentage
sheet.Zoom = 70
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Zoom level";
    //set zoom percentage
    sheet.Zoom = 70;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Zoom level";
    //set zoom percentage
    sheet.Zoom = 70;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Zoom level";
    //set zoom percentage
    sheet.Zoom = 70;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
        "application/msexcel", stream);
    }
}

```

Open a CSV File

A CSV file has to be opened by specifying the delimiter set in it. Specifying the delimiter can be ignored, if it is Comma(,), as Syncfusion XlsIO considers the default delimiter as Comma(,).

The delimiters used in CSV file are Comma (,), Tab (\t), SemiColon (;), Colon (:), Space (), Equals Sign (=) etc..

The following complete code snippet explains how to open a Tab (\t) delimited CSV file using XlsIO.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Open the Tab delimited CSV file
    IWorkbook workbook = application.Workbooks.Open("Sample.csv", "\t");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
'Open the Tab delimited CSV file
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.csv", "\t")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".csv");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Open the Tab delimited CSV file
    IWorkbook workbook = await application.Workbooks.OpenAsync(file, "\t");
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    FileStream inputStream = new FileStream("Sample.csv", FileMode.Open,
    FileAccess.Read);
    //Open the Tab delimited CSV file
    IWorkbook workbook = application.Workbooks.Open(inputStream, "\t");
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    // "App" is the class of Portable project
```

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.csv");
//Open the Tab delimited CSV file
IWorkbook workbook = application.Workbooks.Open(inputStream, "\t");
}

```

Save Worksheet as CSV

The following code example illustrates how to save a worksheet as CSV file.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "document";
    //Save the sheet as CSV
    sheet.SaveAs("Sample.csv", ",");
    workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
sheet.Range("A1:M20").Text = "document"
'Save the sheet as CSV
sheet.SaveAs("Sample.csv", ",")
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "document";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Sample";
    savePicker.FileTypeChoices.Add("CSV Files", new List<string>() { ".csv" });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile1 = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
}

```

```

await sheet.SaveAsAsync(storageFile1, "");
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile2 = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile2);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "document";
    //Saving the sheet and workbook as streams
    FileStream sheetStream = new FileStream("Sample.csv", FileMode.Create,
    FileAccess.ReadWrite);
    sheet.SaveAs(sheetStream, "");
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    sheetStream.Dispose();
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "document";
    //Saving the sheet and workbook as streams
    MemoryStream sheetStream = new MemoryStream();
    sheet.SaveAs(sheetStream, "");
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    sheetStream.Position = 0;
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Sample
        .csv", "application/msexcel", sheetStream);
    }
}

```

```

Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.csv",
"application/msexcel", sheetStream);
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Save worksheet as text (*.txt)

Essential XlsIO allows to save worksheet as a text file. This can be done by leaving the delimiter with a space as shown in the below codes.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Text document";
    //Save the sheet as text
    sheet.SaveAs("Sample.txt", " ");
    workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
sheet.Range("A1:M20").Text = "Text document"
'Save the sheet as text
sheet.SaveAs("Sample.txt", " ")
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Text document";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
}

```

```

savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("Text Files", new List<string>() { ".txt" });
//Creates a storage file from FileSavePicker
StorageFile storageFile1 = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await sheet.SaveAsAsync(storageFile1, " ");
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile2 = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile2);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Text document";
    //Saving the sheet and workbook as streams
    FileStream sheetStream = new FileStream("Sample.txt", FileMode.Create,
    FileAccess.ReadWrite);
    sheet.SaveAs(sheetStream, " ");
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    sheetStream.Dispose();
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Text document";
    //Saving the sheet and workbook as streams
    MemoryStream sheetStream = new MemoryStream();
    sheet.SaveAs(sheetStream, " ");
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    sheetStream.Position = 0;
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
}

```



```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Sample
.txt", "text/plain", sheetStream);
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sample.txt",
"text/plain", sheetStream);
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Save Worksheet as HTML

XlsIO provides support to convert a worksheet or workbook to HTML with basic formatting preserved. The supporting formats in this conversion are:

1. Styles.
2. Hyperlinks.
3. Images.
4. 2D Charts.

The following code example illustrates on how to do this.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
sheet.Range["A1:M20"].Text = "Html Document";
//Save an Excel sheet as HTML file
sheet.SaveAsHtml("Sample.html");
//Save the workbook as HTML file
workbook.SaveAsHtml("Sample.html",
Syncfusion.XlsIO.Implementation.HtmlSaveOptions.Default);
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
sheet.Range("A1:M20").Text = "Html Document"
' Save an Excel sheet as HTML file
sheet.SaveAsHtml("Sample.html")

```

```
'Save a workbook as HTML file
workbook.SaveAsHtml("Sample.html",
Syncfusion.XlsIO.Implementation.HtmlSaveOptions.Default)
End Using
```

UWP

```
//Worksheet To HTML conversion can be performed by referring .NET Standard
assemblies in UWP platform.
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Html Document";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Sample";
    savePicker.FileTypeChoices.Add("HTML Files", new List<string>() {
        ".html", ".htm" });
    //Creates a storage file from the FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Converts and save to stream
    var file = await storageFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream stream = file.AsStreamForWrite();
    //Save an Excel sheet as HTML file
    sheet.SaveAsHtml(stream);
    //Save a workbook as HTML file
    workbook.SaveAsHtml(stream,
    Syncfusion.XlsIO.Implementation.HtmlSaveOptions.Default);
    await file.FlushAsync();
    stream.Dispose();
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Initialize excel engine and open workbook
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A1:M20"].Text = "Html Document";
    //Create stream to store HTML file.
    Stream stream = new MemoryStream();
    //Save an Excel sheet as HTML file
    worksheet.SaveAsHtml(stream);
    //Save a workbook as HTML file
    workbook.SaveAsHtml(stream,
    Syncfusion.XlsIO.Implementation.HtmlSaveOptions.Default);
    stream.Dispose();
    workbook.Close();
}
```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.Range["A1:M20"].Text = "Html Document";
    //Creates memory stream.
    MemoryStream stream = new MemoryStream();
    //Save an Excel sheet as HTML file
    sheet.SaveAsHtml(stream);
    //Save a workbook as HTML file
    workbook.SaveAsHtml(stream,
        Syncfusion.XlsIO.Implementation.HtmlSaveOptions.Default);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
        DependencyService.Get<ISaveWindowsPhone>()
            .Save("Sample.html", "text/html", stream);
    else
        DependencyService.Get<ISave>().Save("Sample.html", "text/html", stream);
}

```

Save Options

XlsIO also provides options to save a worksheet with the displayed text or value in the cell to HTML file. The following code example illustrates this.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create the instant for SaveOptions
    HtmlSaveOptions options = new HtmlSaveOptions();
    options.TextMode = HtmlSaveOptions.GetText.DisplayText;
    options.ImagePath = @"..\..\Images\";
    //Save the sheet as HTML
    sheet.SaveAsHtml("Sample.html", options);
    workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel

```

```

application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Create the instant for SaveOptions
Dim options As New HtmlSaveOptions()
options.TextMode = HtmlSaveOptions.GetText.DisplayText
options.ImagePath = "..\\..\\Images\\"
'Save the sheet as HTML
sheet.SaveAsHtml("Sample.html", options)
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

//Worksheet To HTML conversion can be performed by referring .NET Standard
assemblies in UWP platform.
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Sample";
    savePicker.FileTypeChoices.Add("HTML Files", new List<string>() {
        ".html", ".htm" });
    //Creates a storage file from the FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Create the instant for SaveOptions
    HtmlSaveOptions saveOptions = new HtmlSaveOptions();
    saveOptions.TextMode = HtmlSaveOptions.GetText.DisplayText;
    //Converts and save to stream
    var file = await storageFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream stream = file.AsStreamForWrite();
    sheet.SaveAsHtml(stream, saveOptions);
    await file.FlushAsync();
    stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Initialize excel engine and open workbook
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create stream to store HTML file.
    Stream stream = new MemoryStream();
    //Create the instant for SaveOptions
    HtmlSaveOptions saveOptions = new HtmlSaveOptions();
    saveOptions.TextMode = HtmlSaveOptions.GetText.DisplayText;
    //Save and Dispose
}

```

```
worksheet.SaveAsHtml(stream, saveOptions);
stream.Dispose();
workbook.Close();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create the instant for SaveOptions
    HtmlSaveOptions saveOptions = new HtmlSaveOptions();
    saveOptions.TextMode = HtmlSaveOptions.GetText.DisplayText;
    //Converts and save to stream.
    MemoryStream stream = new MemoryStream();
    sheet.SaveAsHtml(stream, saveOptions);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
        DependencyService.Get<ISaveWindowsPhone>().
            Save("Sample.html", "text/html", stream);
    else
        DependencyService.Get<ISave>().Save("Sample.html", "text/html", stream);
}
```

Worksheet Rows and Columns Manipulation

The Essential XlsIO provides rows and columns manipulation options equivalent to Excel such as insertion, deletion, hiding, adjusting dimensions, grouping, sub-totaling and more through **IWorksheet** interface.

Insert Rows and Columns

The following code snippet illustrates how to insert rows and columns in a worksheet.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Insert a row
    worksheet.InsertRow(3, 1, ExcelInsertOptions.FormatAsBefore);
    //Inserting a column
    worksheet.InsertColumn(2, 1, ExcelInsertOptions.FormatAsAfter);
    workbook.SaveAs("Book1.xlsx");
}
```

```
}

```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Inserting a row
worksheet.InsertRow(3, 1, ExcelInsertOptions.FormatAsBefore)
'Inserting a column
worksheet.InsertColumn(2, 1, ExcelInsertOptions.FormatAsAfter)
workbook.SaveAs("Book1.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Insert a row
    worksheet.InsertRow(3, 1, ExcelInsertOptions.FormatAsBefore);
    //Inserting a column
    worksheet.InsertColumn(2, 1, ExcelInsertOptions.FormatAsAfter);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Book1";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
```

```

IWorkbook workbook = application.Workbooks.Open(fileStream);
IWorksheet worksheet = workbook.Worksheets[0];
//Insert a row
worksheet.InsertRow(3, 1, ExcelInsertOptions.FormatAsBefore);
//Inserting a column
worksheet.InsertColumn(2, 1, ExcelInsertOptions.FormatAsAfter);
//Saving the workbook as stream
FileStream stream = new FileStream("Book1.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Insert a row
    worksheet.InsertRow(3, 1, ExcelInsertOptions.FormatAsBefore);
    //Inserting a column
    worksheet.InsertColumn(2, 1, ExcelInsertOptions.FormatAsAfter);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Book1.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Book1.xlsx", "application/msexcel", stream);
    }
}

```

Note: Row and Column index of Insert methods are "one based".

To know more about insert rows and columns, refer to the **WorksheetImpl** in API section.

Delete Rows and Columns

The following code shows how to delete rows and columns.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Delete a row
    worksheet.DeleteRow(3);
    //Delete a column
    worksheet.DeleteColumn(2);
    workbook.SaveAs("Book1.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Delete a row
worksheet.DeleteRow(3)
'Delete a column
worksheet.DeleteColumn(2)
workbook.SaveAs("Book1.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Delete a row
    worksheet.DeleteRow(3);
    //Delete a column
    worksheet.DeleteColumn(2);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Book1";
}
```



```

savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Delete a row
    worksheet.DeleteRow(3);
    //Delete a column
    worksheet.DeleteColumn(2);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Book1.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    /"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Delete a row
    worksheet.DeleteRow(3);
    //Delete a column
    worksheet.DeleteColumn(2);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)

```

```
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Book1.
xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Book1.xlsx",
"application/msexcel", stream);
}
}
```

Tips: To extract values little faster or to delete a larger number of rows and columns, use Un-Safe code option of **IApplication** interface as follows

C#

```
application.DataProviderType = ExcelDataProviderType.Unsafe;
```

VB.NET

```
application.DataProviderType = ExcelDataProviderType.Unsafe
```

UWP

```
application.DataProviderType = ExcelDataProviderType.Unsafe;
```

ASP.NET CORE

```
//XlsIO supports DataProviderType of IApplication in Windows Forms, WPF and
UWP platforms alone.
```

XAMARIN

```
//XlsIO supports DataProviderType of IApplication in Windows Forms, WPF and
UWP platforms alone.
```

In addition, cells can be deleted by shifting other cells in a row or column towards up/left by one step. This can be done by using the **Clear** method as shown in the following code.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
//Shifts cells towards Left after deletion
worksheet.Range["A1:E1"].Clear(ExcelMoveDirection.MoveLeft);
//Shifts cells toward Up after deletion
worksheet.Range["A1:A6"].Clear(ExcelMoveDirection.MoveUp);
workbook.SaveAs("Book1.xlsx");
}
```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Shifts cells towards Left after deletion
worksheet.Range("A1:E1").Clear(ExcelMoveDirection.MoveLeft)
'Shifts cells towards Up after deletion
worksheet.Range("A1:A6").Clear(ExcelMoveDirection.MoveUp)
workbook.SaveAs("Book1.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Shifts cells towards Left after deletion
    worksheet.Range["A1:E1"].Clear(ExcelMoveDirection.MoveLeft);
    //Shifts cells toward Up after deletion
    worksheet.Range["A1:A6"].Clear(ExcelMoveDirection.MoveUp);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Book1";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Shifts cells towards Left after deletion

```

```

worksheet.Range["A1:E1"].Clear(ExcelMoveDirection.MoveLeft);
//Shifts cells toward Up after deletion
worksheet.Range["A1:A6"].Clear(ExcelMoveDirection.MoveUp);
//Saving the workbook as stream
FileStream stream = new FileStream("Book1.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Shifts cells towards Left after deletion
    worksheet.Range["A1:E1"].Clear(ExcelMoveDirection.MoveLeft);
    //Shifts cells toward Up after deletion
    worksheet.Range["A1:A6"].Clear(ExcelMoveDirection.MoveUp);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Book1.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Book1.xlsx", "application/msexcel", stream);
    }
}

```

Note: Deletion by using above method is more efficient than looping.

Row/Column index of these methods are "one based".

Show or Hide Rows and Columns

Visibility of rows and columns can be set by using the **ShowRow** and **ShowColumn** methods as shown as follows.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Hiding the first column and second row
    worksheet.ShowColumn(1, false);
    worksheet.ShowRow(2, false);
    workbook.SaveAs("Book1.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Hiding the first column and second row
worksheet.ShowColumn(1, False)
worksheet.ShowRow(2, False)
workbook.SaveAs("Book1.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Hiding the first column and second row
    worksheet.ShowColumn(1, false);
    worksheet.ShowRow(2, false);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Book1";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx" });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Hiding the first column and second row
    worksheet.ShowColumn(1, false);
    worksheet.ShowRow(2, false);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Book1.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Hiding the first column and second row
    worksheet.ShowColumn(1, false);
    worksheet.ShowRow(2, false);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Book1.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Book1.xlsx", "application/msexcel", stream);
    }
}
```

```
}
```

Show or Hide Specific Range

The Essential XlsIO allows to set visibility for a specific range. The following code snippet shows how to set the visibility of a range.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    IRange range = worksheet[1, 4];
    //Hiding the range 'D1'
    worksheet.ShowRange(range, false);
    IRange firstRange = worksheet[1, 1, 3, 3];
    IRange secondRange = worksheet[5, 5, 7, 7];
    RangesCollection rangeCollection = new RangesCollection(application,
    worksheet);
    rangeCollection.Add(firstRange);
    rangeCollection.Add(secondRange);
    //Hiding a collection of ranges
    worksheet.ShowRange(rangeCollection, false);
    workbook.SaveAs("Book1.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim range As IRange = worksheet(1, 4)
'Hiding the range 'D1'
worksheet.ShowRange(range, False)
Dim firstRange As IRange = worksheet(1, 1, 3, 3)
Dim secondRange As IRange = worksheet(5, 5, 7, 7)
Dim rangeCollection As RangesCollection = New RangesCollection(application,
worksheet)
rangeCollection.Add(firstRange)
rangeCollection.Add(secondRange)
'Hiding a collection of ranges
worksheet.ShowRange(rangeCollection, False)
workbook.SaveAs("Book1.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
```

```

//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Hiding the first column and second row
worksheet.ShowColumn(1, false);
worksheet.ShowRow(2, false);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Book1";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    IRange range = worksheet[1, 4];
    //Hiding the range 'D1'
    worksheet.ShowRange(range, false);
    IRange firstRange = worksheet[1, 1, 3, 3];
    IRange secondRange = worksheet[5, 5, 7, 7];
    RangesCollection rangeCollection = new RangesCollection(application,
    worksheet);
    rangeCollection.Add(firstRange);
    rangeCollection.Add(secondRange);
    //Hiding a collection of ranges
    worksheet.ShowRange(rangeCollection, false);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Book1.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;

```



```

IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
IRange range = worksheet[1, 4];
//Hiding the range 'D1'
worksheet.ShowRange(range, false);
IRange firstRange = worksheet[1, 1, 3, 3];
IRange secondRange = worksheet[5, 5, 7, 7];
RangesCollection rangeCollection = new RangesCollection(application,
worksheet);
rangeCollection.Add(firstRange);
rangeCollection.Add(secondRange);
//Hiding a collection of ranges
worksheet.ShowRange(rangeCollection, false);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Book1.
xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Book1.xlsx",
"application/msexcel", stream);
}
}

```

Note: Resetting column width manually or through AutoFit, for the columns hidden using [ShowColumn](#)/[HideColumn](#) methods will make the hidden columns visible.

Resetting row height manually or through AutoFit, for the rows hidden using [ShowRow](#) /[HideRow](#) methods will make the hidden rows visible.

Adjust Row Height and Column Width

Rows and columns can be [resized](#) based on its contents. The XlsIO allows to resize rows and columns in the following ways:

- Resize a specific row or column
- Resize a range of rows or columns

Resize a specific row or column

A single row or column can be resized by the **SetRowHeight** and **SetColumnWidth** methods of **IWorksheet**. Similarly, the height and width of a single row or column can be accessed using the **GetRowHeight** or **GetColumnWidth** methods of **IWorksheet**.

The following code snippet shows how to resize a single row and column.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Modifying the row height
    worksheet.SetRowHeight(2, 25);
    //Modifying the column width
    worksheet.SetColumnWidth(1, 20);
    workbook.SaveAs("Book1.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Modifying the row height
worksheet.SetRowHeight(2, 25)
'Modifying the column width
worksheet.SetColumnWidth(1, 20)
workbook.SaveAs("Book1.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Modifying the row height
    worksheet.SetRowHeight(2, 25);
    //Modifying the column width
    worksheet.SetColumnWidth(1, 20);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Book1";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Modifying the row height
    worksheet.SetRowHeight(2, 25);
    //Modifying the column width
    worksheet.SetColumnWidth(1, 20);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Book1.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Modifying the row height
    worksheet.SetRowHeight(2, 25);
    //Modifying the column width
    worksheet.SetColumnWidth(1, 20);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Book1.
        xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Book1.xlsx",
        "application/msexcel", stream);
    }
}

```

Resize a range of rows or columns

Multiple rows or columns can be resized and accessed by using the **RowHeight** and **ColumnWidth** properties of **IRange**. The following code snippet shows how to resize multiple rows and columns.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Modifying the row height
    worksheet.Range["A2:A6"].RowHeight = 25;
    //Modifying the column width
    worksheet.Range["A1:D1"].ColumnWidth = 20;
    workbook.SaveAs("Book1.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Modifying the row height
worksheet.Range("A2:A6").RowHeight = 25
'Modifying the column width
worksheet.Range("A1:D1").ColumnWidth = 20
workbook.SaveAs("Book1.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Modifying the row height
    worksheet.Range["A2:A6"].RowHeight = 25;
    //Modifying the column width
    worksheet.Range["A1:D1"].ColumnWidth = 20;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Book1";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{

```

```

IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Modifying the row height
worksheet.Range["A2:A6"].RowHeight = 25;
//Modifying the column width
worksheet.Range["A1:D1"].ColumnWidth = 20;
//Saving the workbook as stream
FileStream stream = new FileStream("Book1.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Modifying the row height
    worksheet.Range["A2:A6"].RowHeight = 25;
    //Modifying the column width
    worksheet.Range["A1:D1"].ColumnWidth = 20;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Book1.
        xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Book1.xlsx",
        "application/msexcel", stream);
    }
}

```

Note: If a column width or a row height is 0, then the column or row is hidden.

Column width and row height can also be set in pixels, by using the `IWorksheet.SetColumnWidthInPixel` and `IWorksheet.SetRowHeightInPixel` methods respectively.

Auto-Fit Rows and Columns

The XlsIO allows to auto-size the width and height of a cell to fit its content. This section demonstrates various methods to auto-fit rows and columns of a worksheet.

Auto-Fit a Single Row or Column

The following code snippet shows how a row and a column is re-sized to its content.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A1"].Text = "This is a long text";
    worksheet.Range["A1"].WrapText = true;
    //AutoFit applied to a single row
    worksheet.AutofitRow(1);
    worksheet.Range["A3"].Text = "This is a long text";
    //AutoFit applied to a single column
    worksheet.AutofitColumn(3);
    workbook.SaveAs("Book1.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
worksheet.Range("A1").Text = "This is a long text"
worksheet.Range("A1").WrapText = True
'AutoFit applied to a single row
worksheet.AutofitRow(1)
worksheet.Range("A3").Text = "This is a long text"
'AutoFit applied to a single column
worksheet.AutofitColumn(3)
workbook.SaveAs("Book1.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A1"].Text = "This is a long text";
    worksheet.Range["A1"].WrapText = true;
    //AutoFit applied to a single row
    worksheet.AutofitRow(1);
    worksheet.Range["A3"].Text = "This is a long text";
    //AutoFit applied to a single column
}
```

```

worksheet.AutofitColumn(3);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Book1";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A1"].Text = "This is a long text";
    worksheet.Range["A1"].WrapText = true;
    //AutoFit applied to a single row
    worksheet.AutofitRow(1);
    worksheet.Range["A3"].Text = "This is a long text";
    //AutoFit applied to a single column
    worksheet.AutofitColumn(3);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Book1.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A1"].Text = "This is a long text";
    worksheet.Range["A1"].WrapText = true;
    //AutoFit applied to a single row
    worksheet.AutofitRow(1);
    worksheet.Range["A3"].Text = "This is a long text";
    //AutoFit applied to a single column
    worksheet.AutofitColumn(3);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
}

```

```
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Book1.
xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Book1.xlsx",
"application/msexcel", stream);
}
}
```

Note: Row and Column indexes are "one based".

There is an alternative way to auto-fit row or column is by accessing the row or column, which is shown in the following code snippet.

C#

```
//AutoFit applied to first row
worksheet.Rows[0].AutofitRows();
//AutoFit applied to first column
worksheet.Columns[0].AutofitColumns();
```

VB.NET

```
'AutoFit applied to first row
worksheet.Rows(0).AutofitRows()
'AutoFit applied to first column
worksheet.Columns(0).AutofitColumns()
```

UWP

```
//AutoFit applied to first row
worksheet.Rows[0].AutofitRows();
//AutoFit applied to first column
worksheet.Columns[0].AutofitColumns();
```

ASP.NET CORE

```
//AutoFit applied to first row
worksheet.Rows[0].AutofitRows();
//AutoFit applied to first column
worksheet.Columns[0].AutofitColumns();
```

XAMARIN

```
//AutoFit applied to first row
worksheet.Rows[0].AutofitRows();
//AutoFit applied to first column
```



```
worksheet.Columns[0].AutofitColumns();
```

Note: Here column and row indexes are "zero based".

Auto-Fit Multiple Rows or Columns

Multiple rows or columns can be auto fitted based on the range specified. This is depicted in the following code.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Assigning text to cells
    worksheet.Range["A1:D1"].Text = "This is the Long Text";
    worksheet.Range["A2:A5"].Text = "This is the Long Text using AutoFit Columns and Rows";
    worksheet.Range["A2:A5"].WrapText = true;
    //Auto-Fit the range
    worksheet.Range["A1:C1"].AutofitColumns();
    worksheet.Range["A2:A5"].AutofitRows();
    //Auto-fits all the columns used in the worksheet
    worksheet.UsedRange.AutofitColumns();
    workbook.SaveAs("Book1.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Assigning text to cells
worksheet.Range("A1:D1").Text = "This is the Long Text"
worksheet.Range("A2:A5").Text = "This is the Long Text using AutoFit Columns and Rows"
worksheet.Range("A2:A5").WrapText = True
'Auto-Fit the range
worksheet.Range("A1:C1").AutofitColumns()
worksheet.Range("A2:A5").AutofitRows()
'Auto-fits all the columns used in the worksheet
worksheet.UsedRange.AutofitColumns()
workbook.SaveAs("Book1.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
```

```

IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Assigning text to cells
worksheet.Range["A1:D1"].Text = "This is the Long Text";
worksheet.Range["A2:A5"].Text = "This is the Long Text using AutoFit Columns
and Rows";
worksheet.Range["A2:A5"].WrapText = true;
//Auto-Fit the range
worksheet.Range["A1:C1"].AutofitColumns();
worksheet.Range["A2:A5"].AutofitRows();
//Auto-fits all the columns used in the worksheet
worksheet.UsedRange.AutofitColumns();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Book1";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Assigning text to cells
    worksheet.Range["A1:D1"].Text = "This is the Long Text";
    worksheet.Range["A2:A5"].Text = "This is the Long Text using AutoFit Columns
and Rows";
    worksheet.Range["A2:A5"].WrapText = true;
    //Auto-Fit the range
    worksheet.Range["A1:C1"].AutofitColumns();
    worksheet.Range["A2:A5"].AutofitRows();
    //Auto-fits all the columns used in the worksheet
    worksheet.UsedRange.AutofitColumns();
    //Saving the workbook as stream
    FileStream stream = new FileStream("Book1.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
}

```

```

IWorksheet worksheet = workbook.Worksheets[0];
//Assigning text to cells
worksheet.Range["A1:D1"].Text = "This is the Long Text";
worksheet.Range["A2:A5"].Text = "This is the Long Text using AutoFit Columns
and Rows";
worksheet.Range["A2:A5"].WrapText = true;
//Auto-Fit the range
worksheet.Range["A1:C1"].AutofitColumns();
worksheet.Range["A2:A5"].AutofitRows();
//Auto-fits all the columns used in the worksheet
worksheet.UsedRange.AutofitColumns();
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Book1.
xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Book1.xlsx",
"application/msexcel", stream);
}
}

```

Note: 1) If a Range is text wrapped, the AutofitColumn method will not be applied on it.

2) If a Range is merged, the Auto-Fit methods will not be applied on it. Note that this is the behavior of Excel as well.

3) Auto fitting is a time consuming process so, it might cause performance issues when used excessively.

Group or Ungroup Rows and Columns

Rows and columns can be grouped or ungrouped to summarize the data, which is given in the following code snippet.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
//Group Rows
worksheet.Range["A1:A3"].Group(ExcelGroupBy.ByRows, true);
worksheet.Range["A4:A6"].Group(ExcelGroupBy.ByRows);
//Group Columns
worksheet.Range["A1:B1"].Group(ExcelGroupBy.ByColumns, false);
}

```

```

worksheet.Range["C1:F1"].Group(ExcelGroupBy.ByColumns);
//Ungroup Rows
worksheet.Range["A1:A3"].Ungroup(ExcelGroupBy.ByRows);
//Ungroup Columns
worksheet.Range["C1:F1"].Ungroup(ExcelGroupBy.ByColumns);
workbook.SaveAs("Book1.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Group Rows
worksheet.Range("A1:A3").Group(ExcelGroupBy.ByRows, True)
worksheet.Range("A4:A6").Group(ExcelGroupBy.ByRows)
'Group Columns
worksheet.Range("A1:B1").Group(ExcelGroupBy.ByColumns, False)
worksheet.Range("C1:F1").Group(ExcelGroupBy.ByColumns)
'Ungroup Rows
worksheet.Range("A1:A3").Ungroup(ExcelGroupBy.ByRows)
'Ungroup Columns
worksheet.Range("C1:F1").Ungroup(ExcelGroupBy.ByColumns)
workbook.SaveAs("Book1.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Group Rows
worksheet.Range["A1:A3"].Group(ExcelGroupBy.ByRows, true);
worksheet.Range["A4:A6"].Group(ExcelGroupBy.ByRows);
//Group Columns
worksheet.Range["A1:B1"].Group(ExcelGroupBy.ByColumns, false);
worksheet.Range["C1:F1"].Group(ExcelGroupBy.ByColumns);
//Ungroup Rows
worksheet.Range["A1:A3"].Ungroup(ExcelGroupBy.ByRows);
//Ungroup Columns
worksheet.Range["C1:F1"].Ungroup(ExcelGroupBy.ByColumns);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;

```

```

savePicker.SuggestedFileName = "Book1";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Group Rows
    worksheet.Range["A1:A3"].Group(ExcelGroupBy.ByRows, true);
    worksheet.Range["A4:A6"].Group(ExcelGroupBy.ByRows);
    //Group Columns
    worksheet.Range["A1:B1"].Group(ExcelGroupBy.ByColumns, false);
    worksheet.Range["C1:F1"].Group(ExcelGroupBy.ByColumns);
    //Ungroup Rows
    worksheet.Range["A1:A3"].Ungroup(ExcelGroupBy.ByRows);
    //Ungroup Columns
    worksheet.Range["C1:F1"].Ungroup(ExcelGroupBy.ByColumns);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Book1.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    /"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
    ple.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Group Rows
    worksheet.Range["A1:A3"].Group(ExcelGroupBy.ByRows, true);
    worksheet.Range["A4:A6"].Group(ExcelGroupBy.ByRows);
    //Group Columns
    worksheet.Range["A1:B1"].Group(ExcelGroupBy.ByColumns, false);
    worksheet.Range["C1:F1"].Group(ExcelGroupBy.ByColumns);
    //Ungroup Rows

```

```

worksheet.Range["A1:A3"].Ungroup(ExcelGroupBy.ByRows);
//Ungroup Columns
worksheet.Range["C1:F1"].Ungroup(ExcelGroupBy.ByColumns);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Book1.
xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Book1.xlsx",
"application/msexcel", stream);
}
}

```

Expand or Collapse Groups

Groups can be expanded and collapsed using the **ExpandGroups** and **CollapseGroups** methods of **IRange**, which is given in the following code snippet.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
//Expand group with flag set to expand parent
worksheet.Range["A5:A15"].ExpandGroup(ExcelGroupBy.ByRows,
ExpandCollapseFlags.ExpandParent);
//Collapse group
worksheet.Range["A5:A15"].CollapseGroup(ExcelGroupBy.ByRows);
workbook.SaveAs("Book1.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Expand group with flag set to expand parent
worksheet.Range("A5:A15").ExpandGroup(ExcelGroupBy.ByRows,
ExpandCollapseFlags.ExpandParent)
'Collapse group

```

```
worksheet.Range("A5:A15").CollapseGroup(ExcelGroupBy.ByRows)
workbook.SaveAs("Book1.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Expand group with flag set to expand parent
    worksheet.Range["A5:A15"].ExpandGroup(ExcelGroupBy.ByRows,
    ExpandCollapseFlags.ExpandParent);
    //Collapse group
    worksheet.Range["A5:A15"].CollapseGroup(ExcelGroupBy.ByRows);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Book1";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Expand group with flag set to expand parent
    worksheet.Range["A5:A15"].ExpandGroup(ExcelGroupBy.ByRows,
    ExpandCollapseFlags.ExpandParent);
    //Collapse group
    worksheet.Range["A5:A15"].CollapseGroup(ExcelGroupBy.ByRows);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Book1.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

```
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    ///Expand group with flag set to expand parent
    worksheet.Range["A5:A15"].ExpandGroup(ExcelGroupBy.ByRows,
    ExpandCollapseFlags.ExpandParent);
    ///Collapse group
    worksheet.Range["A5:A15"].CollapseGroup(ExcelGroupBy.ByRows);
    ///Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    ///Save the document as file and view the saved document
    ///The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Book1.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Book1.xlsx", "application/msexcel", stream);
    }
}
```

Subtotal

The XlsIO supports subtotalling a group to quickly calculate rows of related data by inserting subtotals and totals.

Various Subtotal options like **Summary below data**, **Replace current subtotals**, **Page break between groups** can be used to customize data.

The following code shows how to add subtotal for a given range.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
```



```

IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
//Set the range for subtotaling
IRange range = worksheet.Range["C3:G12"];
//Perform subtotals for the range with every change in first column
//and subtotals to be included for specified list of columns
range.SubTotal(0, ConsolidationFunction.Sum, new int[] { 2, 3, 4 });
workbook.SaveAs("Book1.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Set the range for subtotaling
Dim range As IRange = worksheet("C3:G12")
'Perform subtotals for the range with every change in first column
'and subtotals to be included for specified list of columns
range.SubTotal(0, ConsolidationFunction.Sum, New Integer() {4})
workbook.SaveAs("Book1.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Set the range for subtotaling
IRange range = worksheet.Range["C3:G12"];
//Perform subtotals for the range with every change in first column
//and subtotals to be included for specified list of columns
range.SubTotal(0, ConsolidationFunction.Sum, new int[] { 2, 3, 4 });
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Book1";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Set the range for subtotaling
    IRange range = worksheet.Range["C3:G12"];
    //Perform subtotals for the range with every change in first column
    //and subtotals to be included for specified list of columns
    range.SubTotal(0, ConsolidationFunction.Sum, new int[] { 2, 3, 4 });
    //Saving the workbook as stream
    FileStream stream = new FileStream("Book1.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Set the range for subtotaling
    IRange range = worksheet.Range["C3:G12"];
    //Perform subtotals for the range with every change in first column
    //and subtotals to be included for specified list of columns
    range.SubTotal(0, ConsolidationFunction.Sum, new int[] { 2, 3, 4 });
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Book1.xlsx", "application/msexcel", stream);
    }
    else

```

```
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Book1.xlsx",
"application/msexcel", stream);
}
}
```

Note: Here column and row indexes are "zero based".

The screenshot of the output with SubTotal generated from the previous code.

	A	B	C	D	E	F	G	H	I
1									
2									
3		Sales Person	Month		Revenue1	Revenue2	Revenue3		
4		Alice	Feb		500	500	500		
5		Alice	Jan		200	200	200		
6		Alice	Mar		2350	2350	2350		
7		Alice Total			3050	3050	3050		
8		Jim	Feb		900	900	900		
9		Jim	Jan		1200	1200	1200		
10		Jim	Mar		1400	1400	1400		
11		Jim Total			3500	3500	3500		
12		Sue	Feb		3600	3600	3600		
13		Sue	Jan		2300	2300	2300		
14		Sue	Mar		1700	1700	1700		
15		Sue Total			7600	7600	7600		
16		Grand Total			14150	14150	14150		
17									
18									
19									

Note: Summary of a group can be shown above the rows and left of the column using the `IsSummaryRowBelow` and `IsSummaryColumnRight` properties of `IPageSetup` interface. By default, these properties are set to `TRUE`.

Worksheet Cells Manipulation

The **IRange** interface represents a single cell or a group of cells in a worksheet. XlsIO has several useful methods for accessing, manipulating and formatting the content in the ranges.

Accessing a Cell or a Range

The following code shows the different ways of accessing a single cell or group of cells

Note: Here row and column indexes in the range are "one based".

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
```

```

IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Access a range by specifying cell address
sheet.Range["A7"].Text = "Accessing a Range by specify cell address ";
//Access a range by specifying cell row and column index
sheet.Range[9, 1].Text = "Accessing a Range by specify cell row and column
index ";
//Access a Range by specifying using defined name
IName name = workbook.Names.Add("Name");
name.RefersToRange = sheet.Range["A11"];
sheet.Range["Name"].Text = "Accessing a Range by specifying using defined
name.";
//Accessing a Range of cells by specifying cells address
sheet.Range["A13:C13"].Text = "Accessing a Range of Cells (Method 1)";
//Accessing a Range of cells specifying cell row and column index
sheet.Range[15, 1, 15, 3].Text = "Accessing a Range of Cells (Method 2)";
workbook.SaveAs("Range.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Access a range by specify cell address
sheet.Range("A7").Text = "Accessing a Range by specify cell address "
'Access a range by specify cell row and column index
sheet.Range(9, 1).Text = "Accessing a Range by specify cell row and column
index "
'Access a Range by specifying using defined name
Dim name As IName = workbook.Names.Add("Name")
name.RefersToRange = sheet.Range("A11")
sheet.Range("Name").Text = "Accessing a Range by specifying using defined
name"
'Accessing a Range of cells by specify cells address
sheet.Range("A13:C13").Text = "Accessing a Range of Cells (Method 1)"
'Accessing a Range of cells specify cell row and column index
sheet.Range(15, 1, 15, 3).Text = "Accessing a Range of Cells (Method 2)"
workbook.SaveAs("Range.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Access a range by specifying cell address
sheet.Range["A7"].Text = "Accessing a Range by specify cell address ";
//Access a range by specifying cell row and column index

```

```

sheet.Range[9, 1].Text = "Accessing a Range by specify cell row and column
index ";
//Access a Range by specifying using defined name
IName name = workbook.Names.Add("Name");
name.RefersToRange = sheet.Range["A11"];
sheet.Range["Name"].Text = "Accessing a Range by specifying using defined
name";
//Accessing a Range of cells by specifying cells address
sheet.Range["A13:C13"].Text = "Accessing a Range of Cells (Method 1)";
//Accessing a Range of cells specifying cell row and column index
sheet.Range[15, 1, 15, 3].Text = "Accessing a Range of Cells (Method 2)";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Range";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Access a range by specifying cell address
    sheet.Range["A7"].Text = "Accessing a Range by specify cell address ";
    //Access a range by specifying cell row and column index
    sheet.Range[9, 1].Text = "Accessing a Range by specify cell row and column
index ";
    //Access a Range by specifying using defined name
    IName name = workbook.Names.Add("Name");
    name.RefersToRange = sheet.Range["A11"];
    sheet.Range["Name"].Text = "Accessing a Range by specifying using defined
name";
    //Accessing a Range of cells by specifying cells address
    sheet.Range["A13:C13"].Text = "Accessing a Range of Cells (Method 1)";
    //Accessing a Range of cells specifying cell row and column index
    sheet.Range[15, 1, 15, 3].Text = "Accessing a Range of Cells (Method 2)";
    //Saving the workbook as stream
    FileStream stream = new FileStream("Range.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{

```

```

IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Access a range by specifying cell address
sheet.Range["A7"].Text = "Accessing a Range by specify cell address ";
//Access a range by specifying cell row and column index
sheet.Range[9, 1].Text = "Accessing a Range by specify cell row and column index ";
//Access a Range by specifying using defined name
IName name = workbook.Names.Add("Name");
name.RefersToRange = sheet.Range["A11"];
sheet.Range["Name"].Text = "Accessing a Range by specifying using defined name";
//Accessing a Range of cells by specifying cells address
sheet.Range["A13:C13"].Text = "Accessing a Range of Cells (Method 1)";
//Accessing a Range of cells specifying cell row and column index
sheet.Range[15, 1, 15, 3].Text = "Accessing a Range of Cells (Method 2)";
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Range.
xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Range.xlsx",
"application/msexcel", stream);
}
}

```

Tips: You can use of GetText, SetText, GetNumber and SetNumber methods from worksheet object that enable users to get/set values without range object.

Accessing Relative Range

By default, accessing a range by index will return the cell or range from worksheet level. To get a relative range for the indexes provided, it is recommended to set the **ExcelRangeIndexerMode** option. Here, the RowIndex and ColumnIndex arguments are relative offsets, where specifying a RowIndex of 1 returns cells in the first row of the range not the first row of the worksheet.

For example, if a range is mentioned as "B3:D5", then accessing a range with the index [1,1] will return the cell "A1" from worksheet. If the **ExcelRangeIndexerMode** is set to **Relative** then it returns "B3".

Following code example illustrates how to access the range relatively to the existing range object in XlsIO.

Note: Here row and column indexes in the range are "one-based".

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Setting range index mode to relative
    application.RangeIndexerMode = ExcelRangeIndexerMode.Relative;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating a range by specifying cells address
    IRange range1 = sheet.Range("B3:D5");
    //Accessing a range relatively to the existing range by specifying cell row
    and column index
    range1[2, 2].Text = "Returns C4 cell";
    range1[0, 0].Text = "Returns A2 cell";
    //Creating a Range of cells specifying cell row and column index
    IRange range2 = sheet.Range[5, 1, 10, 3];
    //Accessing a range relatively to the existing range of cells by specifying
    cell row and column index
    range2[2, 2, 3, 3].Text = "Returns range of cells B6 to C7";
    workbook.SaveAs("Range.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
'Setting range index mode to relative
application.RangeIndexerMode = ExcelRangeIndexerMode.Relative
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creating a range by specifying cells address
Dim range1 As IRange = sheet.Range("B3:D5")
'Accessing a range relatively to the existing range by specifying cell row
and column index
range1(2, 2).Text = "Returns B4 cell"
range1(0, 0).Text = "Returns A2 cell"
Dim range2 As IRange = sheet.Range(5, 1, 10, 3)
'Accessing a range relatively to the existing range of cells by specifying
cell row and column index
range2(2, 2, 3, 3).Text = "Returns range of cells B6 to C7"
workbook.SaveAs("Range.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Setting range index mode to relative
    application.RangeIndexerMode = ExcelRangeIndexerMode.Relative;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
}
```

```

//Creating a range by specifying cells address
IRange range1 = sheet.Range["B3:D5"];
//Accessing a range relatively to the existing range by specifying cell row and column index
range1[2, 2].Text = "Returns C4 cell";
range1[0, 0].Text = "Returns A2 cell";
//Creating a Range of cells specifying cell row and column index
IRange range2 = sheet.Range[5, 1, 10, 3];
//Accessing a range relatively to the existing range of cells by specifying cell row and column index
range2[2, 2, 3, 3].Text = "Returns range of cells B6 to C7";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Range";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx" });
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Setting range index mode to relative
    application.RangeIndexerMode = ExcelRangeIndexerMode.Relative;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating a range by specifying cells address
    IRange range1 = sheet.Range["B3:D5"];
    //Accessing a range relatively to the existing range by specifying cell row and column index
    range1[2, 2].Text = "Returns C4 cell";
    range1[0, 0].Text = "Returns A2 cell";
    //Creating a Range of cells specifying cell row and column index
    IRange range2 = sheet.Range[5, 1, 10, 3];
    //Accessing a range relatively to the existing range of cells by specifying cell row and column index
    range2[2, 2, 3, 3].Text = "Returns range of cells B6 to C7";
    //Saving the workbook as stream
    FileStream stream = new FileStream("Range.xlsx", FileMode.Create, FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;

```



```

application.DefaultVersion = ExcelVersion.Excel2013;
//Setting range index mode to relative
application.RangeIndexerMode = ExcelRangeIndexerMode.Relative;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Creating a range by specifying cells address
IRange range1 = sheet.Range["B3:D5"];
//Accessing a range relatively to the existing range by specifying cell row
and column index
range1[2, 2].Text = "Returns C4 cell";
range1[0, 0].Text = "Returns A2 cell";
//Creating a Range of cells specifying cell row and column index
IRange range2 = sheet.Range[5, 1, 10, 3];
//Accessing a range relatively to the existing range of cells by specifying
cell row and column index
range2[2, 2, 3, 3].Text = "Returns range of cells B6 to C7";
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Range.
xlsx", "application/msexcel", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Range.xlsx",
"application/msexcel", stream);
}
}

```

Accessing Discontinuous Ranges

You can access discontinuous ranges and add them to the **RangesCollection**. You can modify the contents or applying formatting of discontinuous range through RangeCollection instance.

Following code snippet illustrates how to access discontinuous range.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //range1 and range2 are discontinuous ranges
    IRange range1 = sheet.Range["A1:A2"];
    IRange range2 = sheet.Range["C1:C2"];
    IRanges ranges = sheet.CreateRangesCollection();
    //range1 and range2 are considered as a single range
}

```

```

ranges.Add(range1);
ranges.Add(range2);
ranges.Text = "Test";
workbook.SaveAs("Range.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'range1 and range2 are discontinuous ranges
Dim range1 As IRange = sheet.Range("A1:A2")
Dim range2 As IRange = sheet.Range("C1:C2")
Dim ranges As IRanges = sheet.CreateRangesCollection()
'range1 and range2 are considered as a single range
ranges.Add(range1)
ranges.Add(range2)
ranges.Text = "Test"
workbook.SaveAs("Range.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //range1 and range2 are discontinuous ranges
    IRange range1 = sheet.Range["A1:A2"];
    IRange range2 = sheet.Range["C1:C2"];
    IRanges ranges = sheet.CreateRangesCollection();
    //range1 and range2 are considered as a single range
    ranges.Add(range1);
    ranges.Add(range2);
    ranges.Text = "Test";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Range";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())

```

```

{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //range1 and range2 are discontinuous ranges
    IRange range1 = sheet.Range["A1:A2"];
    IRange range2 = sheet.Range["C1:C2"];
    IRanges ranges = sheet.CreateRangesCollection();
    //range1 and range2 are considered as a single range
    ranges.Add(range1);
    ranges.Add(range2);
    ranges.Text = "Test";
    //Saving the workbook as stream
    FileStream stream = new FileStream("Range.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //range1 and range2 are discontinuous ranges
    IRange range1 = sheet.Range["A1:A2"];
    IRange range2 = sheet.Range["C1:C2"];
    IRanges ranges = sheet.CreateRangesCollection();
    //range1 and range2 are considered as a single range
    ranges.Add(range1);
    ranges.Add(range2);
    ranges.Text = "Test";
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Range.
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Range.xlsx",
        "application/msexcel", stream);
    }
}

```

Accessing a Cell or Range using IMigrantRange

The **IMigrantRange** interface can also be used to access a single cell or group of cells and manipulate it. You can prefer IMigrantRange instead of IRange while writing large amount of data which is an optimal way.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    IMigrantRange migrantRange = sheet.MigrantRange;
    //Writing Data
    for (int row = 1; row <= migrantRange.LastRow; row++)
    {
        for (int column = 1; column <= migrantRange.LastColumn; column++)
        {
            //Writing values
            migrantRange.ResetRowColumn(row, column);
            migrantRange.Text = "Test";
        }
    }
    workbook.SaveAs("Range.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim migrantRange As IMigrantRange = sheet.MigrantRange
'Writing Data
Dim row As Integer
For row = 1 To migrantRange.LastRow Step row + 1
Dim column As Integer
For column = 1 To migrantRange.LastColumn Step column + 1
'Writing values
migrantRange.ResetRowColumn(row, column)
migrantRange.Text = "Test"
Next
Next
workbook.SaveAs("Range.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
```

```

IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
IMigrantRange migrantRange = sheet.MigrantRange;
//Writing Data
for (int row = 1; row <= migrantRange.LastRow; row++)
{
    for (int column = 1; column <= migrantRange.LastColumn; column++)
    {
        //Writing values
        migrantRange.ResetRowColumn(row, column);
        migrantRange.Text = "Test";
    }
}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Range";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    IMigrantRange migrantRange = sheet.MigrantRange;
    //Writing Data
    for (int row = 1; row <= migrantRange.LastRow; row++)
    {
        for (int column = 1; column <= migrantRange.LastColumn; column++)
        {
            //Writing values
            migrantRange.ResetRowColumn(row, column);
            migrantRange.Text = "Test";
        }
    }
    //Saving the workbook as stream
    FileStream stream = new FileStream("Range.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;

```

```

application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
IMigrantRange migrantRange = sheet.MigrantRange;
//Writing Data
for (int row = 1; row <= migrantRange.LastRow; row++)
{
    for (int column = 1; column <= migrantRange.LastColumn; column++)
    {
        //Writing values
        migrantRange.ResetRowColumn(row, column);
        migrantRange.Text = "Test";
    }
}
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Range.
    xlsx", "application/msexcel", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Range.xlsx",
    "application/msexcel", stream);
}
}

```

Accessing used range of a Worksheet

The following code snippet shows how to get the range of cells used in a given sheet.

Note: By default, XlsIO considers a cell as used, even if there exists some formatting alone. You can disable this behavior, and make XlsIO consider a cell as used, only when there exists data, by using the `UsedRangeIncludesFormatting` property.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //UsedRange excludes the blank cell which has formatting
    sheet.UsedRangeIncludesFormatting = false;
    //Modifying the column width and row height of the used range
    sheet.UsedRange.ColumnWidth = 20;
    sheet.UsedRange.RowHeight = 20;
    workbook.SaveAs("Range.xlsx");
}

```

```
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'UsedRange excludes the blank cell which has formatting
sheet.UsedRangeIncludesFormatting = False
'Modifying only the Used Ranges
sheet.UsedRange.ColumnWidth = 20
sheet.UsedRange.RowHeight = 20
workbook.SaveAs("Range.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //UsedRange excludes the blank cell which has formatting
    worksheet.UsedRangeIncludesFormatting = false;
    //Modifying the column width and row height of the used range
    worksheet.UsedRange.ColumnWidth = 20;
    worksheet.UsedRange.RowHeight = 20;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Range";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
```

```

FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream);
IWorksheet worksheet = workbook.Worksheets[0];
//UsedRange excludes the blank cell which has formatting
worksheet.UsedRangeIncludesFormatting = false;
//Modifying the column width and row height of the used range
worksheet.UsedRange.ColumnWidth = 20;
worksheet.UsedRange.RowHeight = 20;
//Saving the workbook as stream
FileStream stream = new FileStream("Range.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //UsedRange excludes the blank cell which has formatting
    worksheet.UsedRangeIncludesFormatting = false;
    //Modifying the column width and row height of the used range
    worksheet.UsedRange.ColumnWidth = 20;
    worksheet.UsedRange.RowHeight = 20;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Range.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Range.xlsx", "application/msexcel", stream);
    }
}

```


Get Precedent and Dependent Cells or Range

Precedent cells are cells that are referred to by a formula in another cell. Dependent cells contain formulas that refer to other cells. XlsIO allows to trace the relationship between cells and formulas in Excel workbooks and returns the list of cells or range that are precedent and dependent.

Accessing list of precedent and dependent cells can be obtained:

from a worksheet from a workbook

Following code example illustrates how to get precedent cells from a worksheet and entire workbook.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("FormulaExcel.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Getting precedent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetPrecedents();
    //Getting precedent cells from the workbook
    IRange[] results2 = sheet["A1"].GetPrecedents(true);
    string fileName = "Precedents.xlsx";
    workbook.SaveAs(fileName);
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("FormulaExcel.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Getting precedent cells from the worksheet
Dim results1() As IRange = sheet("A1").GetPrecedents()
'Getting precedent cells from the workbook
Dim results2() As IRange = sheet("A1").GetPrecedents(True)
Dim fileName As String = "Precedents.xlsx"
workbook.SaveAs(fileName)
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet sheet = workbook.Worksheets[0];
    //Getting precedent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetPrecedents();
```

```

//Getting precedent cells from the workbook
IRange[] results2 = sheet["A1"].GetPrecedents(true);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Precedents";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream inputStream = new FileStream("FormulaExcel.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    //Getting precedent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetPrecedents();
    //Getting precedent cells from the workbook
    IRange[] results2 = sheet["A1"].GetPrecedents(true);
    //Saving the workbook as stream
    FileStream file = new FileStream("Precedents.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(file);
    file.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.For
    mulaExcel.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    //Getting precedent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetPrecedents();
    //Getting precedent cells from the workbook
    IRange[] results2 = sheet["A1"].GetPrecedents(true);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
}

```

```
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Preced
ents.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Precedents.xlsx",
"application/msexcel", stream);
}
}
```

Following code example illustrates how to get dependent cells from a worksheet and entire workbook.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("FormulaExcel.xlsx");
IWorksheet sheet = workbook.Worksheets[0];
//Getting dependent cells from the worksheet
IRange[] results1 = sheet["A1"].GetDependents();
//Getting dependent cells from the workbook
IRange[] results2 = sheet["A1"].GetDependents(true);
string fileName = "Dependents.xlsx";
workbook.SaveAs(fileName);
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("FormulaExcel.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Getting dependent cells from the worksheet
Dim results1() As IRange = sheet("A1").GetDependents()
'Getting dependent cells from the workbook
Dim results2() As IRange = sheet("A1").GetDependents(True)
Dim fileName As String = "Dependents.xlsx"
workbook.SaveAs(fileName)
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
```

```

openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet sheet = workbook.Worksheets[0];
//Getting dependent cells from the worksheet
IRange[] results1 = sheet["A1"].GetDependents();
//Getting dependent cells from the workbook
IRange[] results2 = sheet["A1"].GetDependents(true);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Dependents";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream inputStream = new FileStream("FormulaExcel.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    //Getting dependent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetDependents();
    //Getting dependent cells from the workbook
    IRange[] results2 = sheet["A1"].GetDependents(true);
    //Saving the workbook as stream
    FileStream file = new FileStream("Dependents.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(file);
    file.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.For
    mulaExcel.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    //Getting dependent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetDependents();
}

```

```
//Getting dependent cells from the workbook
IRange[] results2 = sheet["A1"].GetDependents(true);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Depend
ents.xlsx", "application/msexcel", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Dependents.xlsx",
"application/msexcel", stream);
}
}
```

Get Direct Precedent and Dependent Cells

GetDirectDependents and **GetDirectPrecedents** methods are used to get direct dependent/precedent cells for source range excluding inner dependent/precedent cells.

Following code example illustrates how to get direct precedent cells from a worksheet and entire workbook.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("FormulaExcel.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Getting precedent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetDirectPrecedents();
    //Getting precedent cells from the workbook
    IRange[] results2 = sheet["A1"].GetDirectPrecedents(true);
    string fileName = "DirectPrecedents.xlsx";
    workbook.SaveAs(fileName);
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("FormulaExcel.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Getting precedent cells from the worksheet
Dim results1() As IRange = sheet("A1").GetDirectPrecedents()
'Getting precedent cells from the workbook
Dim results2() As IRange = sheet("A1").GetDirectPrecedents(True)
```

```
Dim fileName As String = "DirectPrecedents.xlsx"
workbook.SaveAs(fileName)
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet sheet = workbook.Worksheets[0];
    //Getting precedent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetDirectPrecedents();
    //Getting precedent cells from the workbook
    IRange[] results2 = sheet["A1"].GetDirectPrecedents(true);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "DirectPrecedents";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream inputStream = new FileStream("FormulaExcel.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    //Getting precedent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetDirectPrecedents();
    //Getting precedent cells from the workbook
    IRange[] results2 = sheet["A1"].GetDirectPrecedents(true);
    //Saving the workbook as stream
    FileStream file = new FileStream("DirectPrecedents.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(file);
    file.Dispose();
}
```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.For
        mulaExcel.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    ///Getting precedent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetDirectPrecedents();
    ///Getting precedent cells from the workbook
    IRange[] results2 = sheet["A1"].GetDirectPrecedents(true);
    ///Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    ///Save the document as file and view the saved document
    ///The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Direct
        Precedents.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("DirectPrecedents.x
        lsx", "application/msexcel", stream);
    }
}

```

Following code example illustrates how to get direct dependent cells from a worksheet and entire workbook.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("FormulaExcel.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    ///Getting dependent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetDirectDependents();
    ///Getting dependent cells from the workbook
    IRange[] results2 = sheet["A1"].GetDirectDependents(true);
    string fileName = "DirectDependents.xlsx";
    workbook.SaveAs(fileName);
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()

```

```

Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("FormulaExcel.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Getting dependent cells from the worksheet
Dim results1() As IRange = sheet("A1").GetDirectDependents()
'Getting dependent cells from the workbook
Dim results2() As IRange = sheet("A1").GetDirectDependents(True)
Dim fileName As String = "DirectDependents.xlsx"
workbook.SaveAs(fileName)
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet sheet = workbook.Worksheets[0];
    //Getting dependent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetDirectDependents();
    //Getting dependent cells from the workbook
    IRange[] results2 = sheet["A1"].GetDirectDependents(true);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "DirectDependents";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx" });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream inputStream = new FileStream("FormulaExcel.xlsx", FileMode.Open, FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    //Getting dependent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetDirectDependents();
    //Getting dependent cells from the workbook
    IRange[] results2 = sheet["A1"].GetDirectDependents(true);
    //Saving the workbook as stream
}

```



```

FileStream file = new FileStream("DirectDependents.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(file);
file.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.For
        mulaExcel.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    ///Getting dependent cells from the worksheet
    IRange[] results1 = sheet["A1"].GetDirectDependents();
    ///Getting dependent cells from the workbook
    IRange[] results2 = sheet["A1"].GetDirectDependents(true);
    ///Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    ///Save the document as file and view the saved document
    ///The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Direct
        Dependents.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("DirectDependents.x
        lsx", "application/msexcel", stream);
    }
}

```

Clearing a Cell Content

You can delete everything in the cell, or just remove the formatting, contents, comments. The following code example illustrates how to clear a range along with its formatting.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    ///Clearing a Range "A4" and its formatting
}

```

```
sheet.Range["A4"].Clear(true);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("ClearRange.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Clearing a Range "A4" and its formatting
sheet.Range("A4").Clear(True)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("ClearRange.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Clearing a Range "A4" and its formatting
    worksheet.Range["A4"].Clear(true);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "ClearRange";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
}
```

```

IWorksheet worksheet = workbook.Worksheets[0];
//Clearing a Range "A4" and its formatting
worksheet.Range["A4"].Clear(true);
//Saving the workbook as stream
FileStream stream = new FileStream("ClearRange.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Clearing a Range "A4" and its formatting
    worksheet.Range["A4"].Clear(true);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("ClearRange.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ClearRange.xlsx", "application/msexcel", stream);
    }
}

```

Copy or Move a Range

You can copy a range of cells to another range using CopyTo method. You can also copy all the formats or only specific formats using **ExcelCopyRangeOptions** options.

Following code example illustrates how to copy a range of cells from the source to destination.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{

```

```

IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet sheet = workbook.Worksheets[0];
//Copying a Range "A1" to "A5"
IRange source = sheet.Range["A1"];
IRange destination = sheet.Range["A5"];
source.CopyTo(destination, ExcelCopyRangeOptions.All);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("CopyRange.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Copying a Range "A1" to "A5"
Dim source As IRange = sheet.Range("A1")
Dim destination As IRange = sheet.Range("A5")
source.CopyTo(destination, ExcelCopyRangeOptions.All)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("CopyRange.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Copying a Range "A1" to "A5"
IRange source = worksheet.Range["A1"];
IRange destination = worksheet.Range["A5"];
source.CopyTo(destination, ExcelCopyRangeOptions.All);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "CopyRange";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Copying a Range "A1" to "A5"
    IRange source = worksheet.Range["A1"];
    IRange destination = worksheet.Range["A5"];
    source.CopyTo(destination, ExcelCopyRangeOptions.All);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "CopyRange";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    workbook.Version = ExcelVersion.Excel2013;
    await workbook.SaveAsAsync(storageFile);
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Copying a Range "A1" to "A5"
    IRange source = worksheet.Range["A1"];
    IRange destination = worksheet.Range["A5"];
    source.CopyTo(destination, ExcelCopyRangeOptions.All);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
}
```

```
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("CopyRange.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("CopyRange.xlsx", "application/msexcel", stream);
}
}
```

MoveTo method is used for moving a range of cells to another range as shown below.

Note: MoveTo method does not update formulas.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet sheet = workbook.Worksheets[0];
//Moving a Range "A1" to "A5"
IRange source = sheet.Range["A1"];
IRange destination = sheet.Range["A5"];
source.MoveTo(destination);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("MoveRange.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Moving a Range "A1" to "A5"
Dim source As IRange = sheet.Range("A1")
Dim destination As IRange = sheet.Range("A5")
source.MoveTo(destination)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("MoveRange.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
```

```

openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Moving a Range "A1" to "A5"
IRange source = worksheet.Range["A1"];
IRange destination = worksheet.Range["A5"];
source.MoveTo(destination);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "MoveRange";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Moving a Range "A1" to "A5"
    IRange source = worksheet.Range["A1"];
    IRange destination = worksheet.Range["A5"];
    source.MoveTo(destination);
    //Saving the workbook as stream
    FileStream stream = new FileStream("MoveRange.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
    ple.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
}

```

```

IWorksheet worksheet = workbook.Worksheets[0];
//Moving a Range "A1" to "A5"
IRange source = worksheet.Range["A1"];
IRange destination = worksheet.Range["A5"];
source.MoveTo(destination);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("MoveRange.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("MoveRange.xlsx", "application/msexcel", stream);
}
}

```

Copy and Paste As Link

You can copy a range and paste the range as link to another range using a bool parameter in CopyTo method.

Following code example illustrates how to paste a range of cells as link.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet sheet = workbook.Worksheets[0];
//Copy range as link from Range "A1" to "A5"
IRange source = sheet.Range["A1"];
IRange destination = sheet.Range["D5"];
source.CopyTo(destination, true);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("PasteLink.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Copy range as link from Range "A1" to "A5"
Dim source As IRange = sheet.Range("A1")

```



```

Dim destination As IRange = sheet.Range("A5")
source.CopyTo(destination, True)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("PasteLink.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Copy range as link from Range "A1" to "A5"
    IRange source = worksheet.Range["A1"];
    IRange destination = worksheet.Range["A5"];
    source.CopyTo(destination, true);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "PasteLink";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx" });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    workbook.Version = ExcelVersion.Excel2013;
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Copy range as link from Range "A1" to "A5"
    IRange source = worksheet.Range["A1"];
    IRange destination = worksheet.Range["A5"];
    source.CopyTo(destination, true);
    //Saving the workbook as stream
    FileStream stream = new FileStream("PasteLink.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

```
}

```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    ///Copy range as link from Range "A1" to "A5"
    IRange source = worksheet.Range["A1"];
    IRange destination = worksheet.Range["A5"];
    source.CopyTo(destination, true);
    ///Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Position = 0;
    ///Save the document as file and view the saved document
    ///The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("ClearRange.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ClearRange.xlsx", "application/msexcel", stream);
    }
}
```

Skip Blanks While Copying

Blank cells can be skipped while copying from source to destination range by setting the parameter [skip blanks](#) to TRUE.

The following code illustrates how to skip blank cells while copying.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    ///Copy range as link from Range "A1" to "A5".
    IRange source = sheet.Range["A1:A7"];
    IRange destination = sheet.Range["C3"];
```

```
//Skip blanks while copying
source.CopyTo(destination, ExcelCopyRangeOptions.All, true);
//Save workbook
workbook.SaveAs("SkipBlank.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
' Copy range as link from Range "A1" to "A5".
Dim source As IRange = sheet.Range("A1:A7")
Dim destination As IRange = sheet.Range("C3")
' Skip blanks while copying
source.CopyTo(destination, ExcelCopyRangeOptions.All, true)
' Save workbook
workbook.SaveAs("SkipBlank.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker.
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile openFile = await openPicker.PickSingleFileAsync();
    //Opens the workbook.
    IWorkbook workbook = await application.Workbooks.OpenAsync(openFile);
    IWorksheet sheet = workbook.Worksheets[0];
    // Copy range as link from Range "A1" to "A5".
    IRange source = sheet.Range["A1:A7"];
    IRange destination = sheet.Range["C3"];
    //Skip blanks while copying
    source.CopyTo(destination, ExcelCopyRangeOptions.All, true);
    //Initializes FileSavePicker.
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "CreateSpreadsheet";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx" });
    //Creates a storage file from FileSavePicker.
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file.
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
```

```

{
    IApplication application = excelEngine.Excel;
    string basePath = _hostingEnvironment.WebRootPath + @"\"XlsIO\\Sample.xlsx";
    FileStream sampleFile = new FileStream(basePath, FileMode.Open);
    IWorkbook workbook = application.Workbooks.Open(sampleFile);
    IWorksheet sheet = workbook.Worksheets[0];
    // Copy range as link from Range "A1" to "A5".
    IRange source = sheet.Range["A1:A7"];
    IRange destination = sheet.Range["C3"];
    //Skip blanks while copying
    source.CopyTo(destination, ExcelCopyRangeOptions.All, true);
    //Saving the workbook to stream in XLSX format
    FileStream stream = new FileStream("SkipBlank.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    // Copy range as link from Range "A1" to "A5".
    IRange source = sheet.Range["A1:A7"];
    IRange destination = sheet.Range["C3"];
    //Skip blanks while copying
    source.CopyTo(destination, ExcelCopyRangeOptions.All, true);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    workbook.Close();
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("SkipBl
        anks.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("SkipBlanks.xlsx",
        "application/msexcel", stream);
    }
}

```

Find and Replace

You can perform [find and replace](#) text and numbers in workbook or worksheet using XlsIO. Also, XlsIO provides the following options:

To search for data in formulas, values or comments. To search for case-sensitive data and to match entire cell contents of the cell.

To know more about these options, please refer the [ExcelFindType](#), [ExcelFindOptions](#) in the API documentation section.

You can find all the occurrences of a text in worksheet by using [FindAll](#) method. To know more about Find and Replace, please refer [IWorksheet](#) in the API documentation section.

The following code illustrates how to find all the occurrences of text in a worksheet with different find options.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Searches for the given string within the text of worksheet
    IRange[] result1 = sheet.FindAll("FindValue", ExcelFindType.Text);
    //Searches for the given string in formulas
    IRange[] result2 = sheet.FindAll("FindValue", ExcelFindType.Formula);
    //Searches for the given string in calculated value, number and text
    IRange[] result3 = sheet.FindAll("FindValue", ExcelFindType.Values);
    //Searches for the given string in comments
    IRange[] result4 = sheet.FindAll("FindValue", ExcelFindType.Comments);
    //Searches for the given string within the text of worksheet and case
    //matched
    IRange[] result5 = sheet.FindAll("FindValue", ExcelFindType.Text,
    ExcelFindOptions.MatchCase);
    //Searches for the given string within the text of worksheet and the entire
    //cell content matching to search text
    IRange[] result6 = sheet.FindAll("FindValue", ExcelFindType.Text,
    ExcelFindOptions.MatchEntireCellContent);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs("Find.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Searches for the given string within the text of worksheet
Dim result1() As IRange = sheet.FindAll("FindValue", ExcelFindType.Text)
'Searches for the given string in formulas
Dim result2() As IRange = sheet.FindAll("FindValue", ExcelFindType.Formula)
'Searches for the given string in calculated value, number and text
Dim result3() As IRange = sheet.FindAll("FindValue", ExcelFindType.Values)
'Searches for the given string in comments
Dim result4() As IRange = sheet.FindAll("FindValue", ExcelFindType.Comments)
```

```

'Searches for the given string within the text of worksheet and case matched
Dim result5() As IRange = sheet.FindAll("FindValue", ExcelFindType.Text,
ExcelFindOptions.MatchCase)
'Searches for the given string within the text of worksheet and the entire
cell content matching to search text
Dim result6() As IRange = sheet.FindAll("FindValue", ExcelFindType.Text,
ExcelFindOptions.MatchEntireCellContent)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Find.xlsx")
End Using

```

UWP

```

ExcelEngine excelEngine = new ExcelEngine();
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Searches for the given string within the text of worksheet
    IRange[] result1 = worksheet.FindAll("FindValue", ExcelFindType.Text);
    //Searches for the given string in formulas
    IRange[] result2 = worksheet.FindAll("FindValue", ExcelFindType.Formula);
    //Searches for the given string in calculated value, number and text
    IRange[] result3 = worksheet.FindAll("FindValue", ExcelFindType.Values);
    //Searches for the given string in comments
    IRange[] result4 = worksheet.FindAll("FindValue", ExcelFindType.Comments);
    //Searches for the given string within the text of worksheet and case
matched
    IRange[] result5 = worksheet.FindAll("FindValue", ExcelFindType.Text,
ExcelFindOptions.MatchCase);
    //Searches for the given string within the text of worksheet and the entire
cell content matching to search text
    IRange[] result6 = worksheet.FindAll("FindValue", ExcelFindType.Text,
ExcelFindOptions.MatchEntireCellContent);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Find";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    workbook.Version = ExcelVersion.Excel2013;
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Searches for the given string within the text of worksheet
    IRange[] result1 = worksheet.FindAll("FindValue", ExcelFindType.Text);
    //Searches for the given string in formulas
    IRange[] result2 = worksheet.FindAll("FindValue", ExcelFindType.Formula);
    //Searches for the given string in calculated value, number and text
    IRange[] result3 = worksheet.FindAll("FindValue", ExcelFindType.Values);
    //Searches for the given string in comments
    IRange[] result4 = worksheet.FindAll("FindValue", ExcelFindType.Comments);
    //Searches for the given string within the text of worksheet and case
    matched
    IRange[] result5 = worksheet.FindAll("FindValue", ExcelFindType.Text,
    ExcelFindOptions.MatchCase);
    //Searches for the given string within the text of worksheet and the entire
    cell content matching to search text
    IRange[] result6 = worksheet.FindAll("FindValue", ExcelFindType.Text,
    ExcelFindOptions.MatchEntireCellContent);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Find.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
    ple.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Searches for the given string within the text of worksheet
    IRange[] result1 = worksheet.FindAll("FindValue", ExcelFindType.Text);
    //Searches for the given string in formulas
    IRange[] result2 = worksheet.FindAll("FindValue", ExcelFindType.Formula);
    //Searches for the given string in calculated value, number and text
    IRange[] result3 = worksheet.FindAll("FindValue", ExcelFindType.Values);
    //Searches for the given string in comments
    IRange[] result4 = worksheet.FindAll("FindValue", ExcelFindType.Comments);
    //Searches for the given string within the text of worksheet and case
    matched
}

```

```

IRange[] result5 = worksheet.FindAll("FindValue", ExcelFindType.Text,
ExcelFindOptions.MatchCase);
//Searches for the given string within the text of worksheet and the entire
cell content matching to search text
IRange[] result6 = worksheet.FindAll("FindValue", ExcelFindType.Text,
ExcelFindOptions.MatchEntireCellContent);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Find.x
lsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Find.xlsx",
"application/msexcel", stream);
}
}

```

You can replace a text with another text with the help of Replace method which searches for text you'd like to change. You can replace a string, with the data of various data types and data sources, such as data table, data column and array.

To know more about replace overloads, please refer [Replace](#) in the API documentation section.

The following code example illustrates how to replace all occurrences of given string with various data.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet sheet = workbook.Worksheets[0];
//Replaces the given string with another string
sheet.Replace("FindValue", "NewValue");
//Replaces the given string with another string on match case
sheet.Replace("FindValue", "NewValue", ExcelFindOptions.MatchCase);
//Replaces the given string with another string matching entire cell content
to the search word
sheet.Replace("FindValue", "NewValue",
ExcelFindOptions.MatchEntireCellContent);
//Replaces the given string with DateTime value
sheet.Replace("DateValue", DateTime.Now);
//Replaces the given string with Array
sheet.Replace("ArrayValue", new string[] { "ArrayValue1", "ArrayValue2",
"ArrayValue3" }, true);
//Replaces the given string with DataTable

```



```

DataTable table = SampleDataTable();
sheet.Replace("DataTable", table, true);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("Replace.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Replaces the given string with another string
sheet.Replace("FindValue", "NewValue")
'Replaces the given string with another string on match case
sheet.Replace("FindValue", "NewValue", ExcelFindOptions.MatchCase)
'Replaces the given string with another string matching entire cell content
to the search word
sheet.Replace("FindValue", "NewValue",
ExcelFindOptions.MatchEntireCellContent)
'Replaces the given string with DateTime value
sheet.Replace("DateValue", DateTime.Now)
'Replaces the given string with Array
sheet.Replace("ArrayValue", New String() {"ArrayValue1", "ArrayValue2",
"ArrayValue3"}, True)
'Replaces the given string with DataTable
Dim table As DataTable = SampleDataTable()
sheet.Replace("DataTable", table, True)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Replace.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Replaces the given string with another string
worksheet.Replace("FindValue", "NewValue");
//Replaces the given string with another string on match case
worksheet.Replace("FindValue", "NewValue", ExcelFindOptions.MatchCase);
//Replaces the given string with another string matching entire cell content
to the search word
worksheet.Replace("FindValue", "NewValue",
ExcelFindOptions.MatchEntireCellContent);
//Replaces the given string with DateTime value
worksheet.Replace("DateValue", DateTime.Now);
}

```

```

//Replaces the given string with Array
worksheet.Replace("ArrayValue", new string[] { "ArrayValue1", "ArrayValue2",
"ArrayValue3" }, true);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Replace";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Replaces the given string with another string
    worksheet.Replace("FindValue", "NewValue");
    //Replaces the given string with another string on match case
    worksheet.Replace("FindValue", "NewValue", ExcelFindOptions.MatchCase);
    //Replaces the given string with another string matching entire cell content
    to the search word
    worksheet.Replace("FindValue", "NewValue",
    ExcelFindOptions.MatchEntireCellContent);
    //Replaces the given string with DateTime value
    worksheet.Replace("DateValue", DateTime.Now);
    //Replaces the given string with Array
    worksheet.Replace("ArrayValue", new string[] { "ArrayValue1", "ArrayValue2",
    "ArrayValue3" }, true);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Replace.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    /"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
    ple.xlsx");
}

```

```

IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet worksheet = workbook.Worksheets[0];
//Replaces the given string with another string
worksheet.Replace("FindValue", "NewValue");
//Replaces the given string with another string on match case
worksheet.Replace("FindValue", "NewValue", ExcelFindOptions.MatchCase);
//Replaces the given string with another string matching entire cell content
to the search word
worksheet.Replace("FindValue", "NewValue",
ExcelFindOptions.MatchEntireCellContent);
//Replaces the given string with DateTime value
worksheet.Replace("DateValue", DateTime.Now);
//Replaces the given string with Array
worksheet.Replace("ArrayValue", new string[] { "ArrayValue1", "ArrayValue2",
"ArrayValue3" }, true);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Replac
e.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Replace.xlsx",
"application/msexcel", stream);
}
}

```

Data Sorting

You can sort a range of cells based on data in one or more columns. You can perform sorting based on the following:

- Based on Cell Values
- Based on Font Color
- Based on Cell Color

Note: Currently XlsIO don't support sorting based on cell icon, parsing and serialization of its sorting details.

Based on Cell Values

The following code snippet explains how to sort a range of cells by values

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
```

```

{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Creates the data sorter
    IDataSort sorter = workbook.CreateDataSorter();
    //Range to sort
    sorter.SortRange = sheet.Range["D3:D16"];
    //Adds the sort field with the column index, sort based on and order by attribute
    ISortField sortField = sorter.SortFields.Add(0, SortOn.Values,
    OrderBy.Ascending);
    //Adds another sort field
    ISortField sortField2 = sorter.SortFields.Add(1, SortOn.Values,
    OrderBy.Ascending);
    //Sort based on the sort Field attribute
    sorter.Sort();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs("Sort.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creates the Data sorter
Dim sorter As IDataSort = workbook.CreateDataSorter()
'Specifies the sort range
sorter.SortRange = sheet.Range("D3:D16")
'Adds the sort field with column index, sort based on and order by attribute
Dim sortField As ISortField = sorter.SortFields.Add(0, SortOn.Values,
OrderBy.Ascending)
'Adds the second sort field
Dim sortField2 As ISortField = sorter.SortFields.Add(1, SortOn.Values,
OrderBy.Ascending)
'Sorts the data with the sort field attribute
sorter.Sort()
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Sort.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
}

```

```
IWorksheet worksheet = workbook.Worksheets[0];
//Creates the data sorter
IDataSort sorter = workbook.CreateDataSorter();
//Range to sort
sorter.SortRange = worksheet.Range["D3:D16"];
//Adds the sort field with the column index, sort based on and order by
attribute
ISortField sortField = sorter.SortFields.Add(0, SortOn.Values,
OrderBy.Ascending);
//Adds another sort field
ISortField sortField2 = sorter.SortFields.Add(1, SortOn.Values,
OrderBy.Ascending);
//Sort based on the sort Field attribute
sorter.Sort();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sort";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("SortingData.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creates the data sorter
    IDataSort sorter = workbook.CreateDataSorter();
    //Range to sort
    sorter.SortRange = worksheet.Range["D3:D16"];
    //Adds the sort field with the column index, sort based on and order by
    attribute
    ISortField sortField = sorter.SortFields.Add(0, SortOn.Values,
    OrderBy.Ascending);
    //Adds another sort field
    ISortField sortField2 = sorter.SortFields.Add(1, SortOn.Values,
    OrderBy.Ascending);
    //Sort based on the sort Field attribute
    sorter.Sort();
    //Saving the workbook as stream
    FileStream stream = new FileStream("Sort.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    /"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creates the data sorter
    IDataSort sorter = workbook.CreateDataSorter();
    //Range to sort
    sorter.SortRange = worksheet.Range["D3:D16"];
    //Adds the sort field with the column index, sort based on and order by attribute
    ISortField sortField = sorter.SortFields.Add(0, SortOn.Values,
        OrderBy.Ascending);
    //Adds another sort field
    ISortField sortField2 = sorter.SortFields.Add(1, SortOn.Values,
        OrderBy.Ascending);
    //Sort based on the sort Field attribute
    sorter.Sort();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer xlsio/xamarin section for respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Sort.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sort.xlsx", "application/msexcel", stream);
    }
}

```

Based on Font Color

The following code snippet explains how to move a range of cells with the specified font color to either top or bottom of the sorting range.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{

```

```

IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet sheet = workbook.Worksheets[0];
//Creates the data sorter
IDataSort sorter = workbook.CreateDataSorter();
//Range to sort
sorter.SortRange = sheet.Range["A2:D16"];
//Creates the sort field with the column index, sort based on and order by attribute
ISortField sortField1 = sorter.SortFields.Add(2, SortOn.FontColor, OrderBy.OnTop);
//Specifies the color to sort the data
sortField1.Color = Color.Red;
//Creates another sort field with the column index, sort based on and order by attribute
ISortField sortField2 = sorter.SortFields.Add(2, SortOn.FontColor, OrderBy.OnTop);
//Specifies the color to sort the data
sortField2.Color = Color.Green;
//Sort based on the sort Field attribute
sorter.Sort();
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("Sort.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creates the Data sorter
Dim sorter As IDataSort = workbook.CreateDataSorter()
'Specifies the sort range
sorter.SortRange = sheet.Range("A2:D16")
'Adds the sort field with column index, sort based on and order by attribute
Dim field1 As ISortField = sorter.SortFields.Add(2, SortOn.FontColor, OrderBy.OnTop)
'Sorts the data based on this color
field1.Color = Color.Red
'Adds another sort field with column index, sort based on and order by attribute
Dim field2 As ISortField = sorter.SortFields.Add(2, SortOn.FontColor, OrderBy.OnTop)
'Sorts the data based on this color
field2.Color = Color.Green
'Sorts the data with the sort field attribute
sorter.Sort()
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Sort.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{

```

```

IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Creates the data sorter
IDataSort sorter = workbook.CreateDataSorter();
//Range to sort
sorter.SortRange = worksheet.Range["A2:D16"];
//Creates the sort field with the column index, sort based on and order by
attribute
ISortField sortField1 = sorter.SortFields.Add(2, SortOn.FontColor,
OrderBy.OnTop);
//Specifies the color to sort the data
sortField1.Color = Color.FromArgb(255, 255, 0, 0);
//Creates another sort field with the column index, sort based on and order
by attribute
ISortField sortField2 = sorter.SortFields.Add(2, SortOn.FontColor,
OrderBy.OnTop);
//Specifies the color to sort the data
sortField2.Color = Color.FromArgb(255, 0, 128, 0);
//Sort based on the sort Field attribute
sorter.Sort();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sort";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream);
IWorksheet worksheet = workbook.Worksheets[0];
//Creates the data sorter
IDataSort sorter = workbook.CreateDataSorter();
//Range to sort
sorter.SortRange = worksheet.Range["A2:D16"];
//Creates the sort field with the column index, sort based on and order by
attribute

```



```

ISortField sortField1 = sorter.SortFields.Add(2, SortOn.FontColor,
OrderBy.OnTop);
//Specifies the color to sort the data
sortField1.Color = Color.Red;
//Creates another sort field with the column index, sort based on and order
by attribute
ISortField sortField2 = sorter.SortFields.Add(2, SortOn.FontColor,
OrderBy.OnTop);
//Specifies the color to sort the data
sortField2.Color = Color.Green;
//Sort based on the sort Field attribute
sorter.Sort();
//Saving the workbook as stream
FileStream stream = new FileStream("Sort.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
// "App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet worksheet = workbook.Worksheets[0];
//Creates the data sorter
IDataSort sorter = workbook.CreateDataSorter();
//Range to sort
sorter.SortRange = worksheet.Range["A2:D16"];
//Creates the sort field with the column index, sort based on and order by
attribute
ISortField sortField1 = sorter.SortFields.Add(2, SortOn.FontColor,
OrderBy.OnTop);
//Specifies the color to sort the data
sortField1.Color = Syncfusion.Drawing.Color.Red;
//Creates another sort field with the column index, sort based on and order
by attribute
ISortField sortField2 = sorter.SortFields.Add(2, SortOn.FontColor,
OrderBy.OnTop);
//Specifies the color to sort the data
sortField2.Color = Syncfusion.Drawing.Color.Green;
//Sort based on the sort Field attribute
sorter.Sort();
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
}

```

```
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Sort.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sort.xlsx", "application/msexcel", stream);
}
}
```

Based on Cell Color

The following code snippet explains how to move a range of cells with the specified cell background color to either top or bottom of the sorting range.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet sheet = workbook.Worksheets[0];
//Creates the data sorter
IDataSort sorter = workbook.CreateDataSorter();
//Range to sort
sorter.SortRange = sheet.Range["A2:D16"];
//Creates the sort field with the column index, sort based on and order by
attribute
ISortField sortField1 = sorter.SortFields.Add(2, SortOn.CellColor,
OrderBy.OnTop);
//Specifies the color to sort the data
sortField1.Color = Color.Red;
//Creates the sort field with the column index, sort based on and order by
attribute
ISortField sortField2 = sorter.SortFields.Add(2, SortOn.CellColor,
OrderBy.OnTop);
//Specifies the color to sort the data
sortField2.Color = Color.Green;
//Sort based on the sort field attribute
sorter.Sort();
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("Sort.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
```

```

Dim sorter As IDataSort = workbook.CreateDataSorter()
'Specifies the sort range.
sorter.SortRange = sheet.Range("A2:D16")
'Adds the sort field with column index, sort based on and order by attribute
Dim field1 As ISortField = sorter.SortFields.Add(2, SortOn.CellColor,
OrderBy.OnTop)
'Sorts the data based on this color
field1.Color = Color.Red
'Adds the sort field with column index, sort based on and order by attribute
Dim field2 As ISortField = sorter.SortFields.Add(2, SortOn.CellColor,
OrderBy.OnTop)
'Sorts the data based on this color
field2.Color = Color.Green
'Sorts the data with the sort field attribute
sorter.Sort()
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Sort.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creates the data sorter
    IDataSort sorter = workbook.CreateDataSorter();
    //Range to sort
    sorter.SortRange = worksheet.Range["A2:D16"];
    //Creates the sort field with the column index, sort based on and order by
    attribute
    ISortField sortField1 = sorter.SortFields.Add(2, SortOn.CellColor,
    OrderBy.OnTop);
    //Specifies the color to sort the data
    sortField1.Color = Color.FromArgb(255, 255, 0, 0);
    //Creates another sort field with the column index, sort based on and order
    by attribute
    ISortField sortField2 = sorter.SortFields.Add(2, SortOn.CellColor,
    OrderBy.OnTop);
    //Specifies the color to sort the data
    sortField2.Color = Color.FromArgb(255, 0, 128, 0);
    //Sort based on the sort Field attribute
    sorter.Sort();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Sort";
}

```

```

savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creates the data sorter
    IDataSort sorter = workbook.CreateDataSorter();
    //Range to sort
    sorter.SortRange = worksheet.Range["A2:D16"];
    //Creates the sort field with the column index, sort based on and order by
attribute
    ISortField sortField1 = sorter.SortFields.Add(2, SortOn.CellColor,
    OrderBy.OnTop);
    //Specifies the color to sort the data
    sortField1.Color = Color.Red;
    //Creates another sort field with the column index, sort based on and order
by attribute
    ISortField sortField2 = sorter.SortFields.Add(2, SortOn.CellColor,
    OrderBy.OnTop);
    //Specifies the color to sort the data
    sortField2.Color = Color.Green;
    //Sort based on the sort Field attribute
    sorter.Sort();
    //Saving the workbook as stream
    FileStream stream = new FileStream("Sort.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
    ple.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
}

```

```

//Creates the data sorter
IDataSort sorter = workbook.CreateDataSorter();
//Range to sort
sorter.SortRange = worksheet.Range["A2:D16"];
//Creates the sort field with the column index, sort based on and order by attribute
ISortField sortField1 = sorter.SortFields.Add(2, SortOn.CellColor,
OrderBy.OnTop);
//Specifies the color to sort the data
sortField1.Color = Syncfusion.Drawing.Color.Red;
//Creates another sort field with the column index, sort based on and order by attribute
ISortField sortField2 = sorter.SortFields.Add(2, SortOn.CellColor,
OrderBy.OnTop);
//Specifies the color to sort the data
sortField2.Color = Syncfusion.Drawing.Color.Green;
//Sort based on the sort Field attribute
sorter.Sort();
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer xlsio/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Sort.xlsx", "application/msexcel", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sort.xlsx", "application/msexcel", stream);
}
}

```

Data Filtering

Using [AutoFilters](#), you can filter data to enable quick and easy way to find and work with a subset of data in a range of cells. When you filter data, entire rows are hidden if values in one or more columns don't meet the filtering criteria. The following are the types of filters that can be used in XlsIO.

- Custom Filter (Conditional)
- Combination Filter (Text and DateTime filter)
- Dynamic Filter
- Color Filter
- Icon Filter
- Advanced Filter

Applying Filter

The following code illustrates how to apply simple auto filters.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
    range.
    sheet.AutoFilters.FilterRange = sheet.Range["A1:K180"];
    //Column index to which AutoFilter must be applied
    IAutoFilter filter = sheet.AutoFilters[0];
    //To apply Top10Number filter, IsTop and IsTop10 must be enabled
    filter.IsTop = true;
    filter.IsTop10 = true;
    //Setting Top10 filter with number of cell to be filtered from top
    filter.Top10Number = 5;
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs("Filter.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
range.
sheet.AutoFilters.FilterRange = sheet.Range("A1:K180")
'Column index to which AutoFilter must be applied.
Dim filter As IAutoFilter = sheet.AutoFilters(0)
'To apply Top10Number filter, IsTop and IsTop10 must be enabled.
filter.IsTop = True
filter.IsTop10 = True
'Setting Top10 filter with number of cell to be filtered from top
filter.Top10Number = 5
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Filter.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
}
```

```

IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
range
worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
//Column index to which AutoFilter must be applied
IAutoFilter filter = worksheet.AutoFilters[0];
//To apply Top10Number filter, IsTop and IsTop10 must be enabled
filter.IsTop = true;
filter.IsTop10 = true;
//Setting Top10 filter with number of cell to be filtered from top
filter.Top10Number = 5;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Filter";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream);
IWorksheet worksheet = workbook.Worksheets[0];
//Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
range
worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
//Column index to which AutoFilter must be applied
IAutoFilter filter = worksheet.AutoFilters[0];
//To apply Top10Number filter, IsTop and IsTop10 must be enabled
filter.IsTop = true;
filter.IsTop10 = true;
//Setting Top10 filter with number of cell to be filtered from top
filter.Top10Number = 5;
//Saving the workbook as stream
FileStream stream = new FileStream("Filter.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{

```

```

IApplication application = excelEngine.Excel;
// "App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet worksheet = workbook.Worksheets[0];
// Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range
worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
// Column index to which AutoFilter must be applied
IAutoFilter filter = worksheet.AutoFilters[0];
// To apply Top10Number filter, IsTop and IsTop10 must be enabled
filter.IsTop = true;
filter.IsTop10 = true;
// Setting Top10 filter with number of cell to be filtered from top
filter.Top10Number = 5;
// Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Position = 0;
// Save the document as file and view the saved document
// The operation in SaveAndView under Xamarin varies between Windows Phone,
// Android and iOS platforms. Please refer xlsio/xamarin section for respective
// code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Filter.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Filter.xlsx", "application/msexcel", stream);
}
}

```

Custom Filter

Following code snippets illustrates how to apply custom filter, based on first and second condition.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet sheet = workbook.Worksheets[0];
// Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range
sheet.AutoFilters.FilterRange = sheet.Range["A1:B323"];
IAutoFilter filter = sheet.AutoFilters[1];
// Specifying first condition
IAutoFilterCondition firstCondition = filter.FirstCondition;

```



```

firstCondition.ConditionOperator = ExcelFilterCondition.Greater;
firstCondition.Double = 100;
//Specifying second condition
IAutoFilterCondition secondCondition = filter.SecondCondition;
secondCondition.ConditionOperator = ExcelFilterCondition.Less;
secondCondition.Double = 200;
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("Filter.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range.
sheet.AutoFilters.FilterRange = sheet.Range("A1:B323")
Dim filter As IAutoFilter = sheet.AutoFilters(1)
'Specifying first condition.
Dim firstCondition As IAutoFilterCondition = filter.FirstCondition
firstCondition.ConditionOperator = ExcelFilterCondition.Greater
firstCondition.Double = 100
'Specifying second condition.
Dim secondCondition As IAutoFilterCondition = filter.SecondCondition
secondCondition.ConditionOperator = ExcelFilterCondition.Less
secondCondition.Double = 200
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Filter.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet sheet = workbook.Worksheets[0];
//Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range
sheet.AutoFilters.FilterRange = sheet.Range["A1:B323"];
IAutoFilter filter = sheet.AutoFilters[1];
//Specifying first condition
IAutoFilterCondition firstCondition = filter.FirstCondition;
firstCondition.ConditionOperator = ExcelFilterCondition.Greater;
firstCondition.Double = 100;
//Specifying second condition
IAutoFilterCondition secondCondition = filter.SecondCondition;

```

```

secondCondition.ConditionOperator = ExcelFilterCondition.Less;
secondCondition.Double = 200;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Filter";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
    range
    sheet.AutoFilters.FilterRange = sheet.Range["A1:B323"];
    IAutoFilter filter = sheet.AutoFilters[1];
    //Specifying first condition
    IAutoFilterCondition firstCondition = filter.FirstCondition;
    firstCondition.ConditionOperator = ExcelFilterCondition.Greater;
    firstCondition.Double = 100;
    //Specifying second condition
    IAutoFilterCondition secondCondition = filter.SecondCondition;
    secondCondition.ConditionOperator = ExcelFilterCondition.Less;
    secondCondition.Double = 200;
    //Saving the workbook as stream
    FileStream file = new FileStream("Filter.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(file);
    file.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
    ple.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
}

```

```

//Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range
sheet.AutoFilters.FilterRange = sheet.Range["A1:B323"];
IAutoFilter filter = sheet.AutoFilters[1];
//Specifying first condition
IAutoFilterCondition firstCondition = filter.FirstCondition;
firstCondition.ConditionOperator = ExcelFilterCondition.Greater;
firstCondition.Double = 100;
//Specifying second condition
IAutoFilterCondition secondCondition = filter.SecondCondition;
secondCondition.ConditionOperator = ExcelFilterCondition.Less;
secondCondition.Double = 200;
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer xlsio/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Filter.xlsx", "application/msexcel", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Filter.xlsx", "application/msexcel", stream);
}
}

```

Combination Filter

This filter contains both Text filter and DateTime filter, it filters the data based on multiple criteria. Following code snippets illustrates how to apply combination filter with multiple of Text filter and DateTime filter.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range
    sheet.AutoFilters.FilterRange = sheet.Range["A1:K180"];
    //Column index to which AutoFilter must be applied
    IAutoFilter filter = sheet.AutoFilters[2];
    //Applying Text filter to filter multiple text to get filter
    filter.AddTextFilter(new string[] { "London", "Paris", "New York City" });
    //Applying DateTime filter to filter the date based on DateTimeGroupingType
    filter.AddDateFilter(new DateTime(2013, 1, 29, 0, 0, 0), DateTimeGroupingType.day);
}

```

```
filter.AddDateFilter(2014, 12, 2, 10, 30, 0, DateTimeGroupingType.minute);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("Filter.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
range
sheet.AutoFilters.FilterRange = sheet.Range("A1:K180")
'Column index to which AutoFilter must be applied
Dim filter As IAutoFilter = sheet.AutoFilters(2)
'Applying Text filter to filter multiple text to get filter
filter.AddTextFilter(New String() {"London", "Paris", "New York City"})
'Applying DateTime filter to filter the date based on DateTimeGroupingType
filter.AddDateFilter(New DateTime(2013, 1, 29, 0, 0, 0),
DateTimeGroupingType.day)
filter.AddDateFilter(2014, 12, 2, 10, 30, 0, DateTimeGroupingType.minute)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Filter.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
range.
worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
//Column index to which AutoFilter must be applied.
IAutoFilter filter = worksheet.AutoFilters[2];
//Applying Text filter to filter multiple text to get filter.
filter.AddTextFilter(new string[] { "London", "Paris", "New York City" });
//Applying DateTime filter to filter the date based on DateTimeGroupingType.
filter.AddDateFilter(new DateTime(2013, 1, 29, 0, 0, 0),
DateTimeGroupingType.day);
filter.AddDateFilter(2014, 12, 2, 10, 30, 0, DateTimeGroupingType.minute);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Filter";
```

```

savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
    range.
    worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
    //Column index to which AutoFilter must be applied.
    IAutoFilter filter = worksheet.AutoFilters[2];
    //Applying Text filter to filter multiple text to get filter.
    filter.AddTextFilter(new string[] { "London", "Paris", "New York City" });
    //Applying DateTime filter to filter the date based on DateTimeGroupingType.
    filter.AddDateFilter(new DateTime(2013, 1, 29, 0, 0, 0),
    DateTimeGroupingType.day);
    filter.AddDateFilter(2014, 12, 2, 10, 30, 0, DateTimeGroupingType.minute);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Filter.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    /"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
    ple.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
    range.
    worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
    //Column index to which AutoFilter must be applied.
    IAutoFilter filter = worksheet.AutoFilters[2];
    //Applying Text filter to filter multiple text to get filter.
    filter.AddTextFilter(new string[] { "London", "Paris", "New York City" });
}

```

```
//Applying DateTime filter to filter the date based on DateTimeGroupingType.
filter.AddDateFilter(new DateTime(2013, 1, 29, 0, 0, 0),
DateTimeGroupingType.day);
filter.AddDateFilter(2014, 12, 2, 10, 30, 0, DateTimeGroupingType.minute);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Filter
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Filter.xlsx",
"application/msexcel", stream);
}
}
```

Dynamic Filter

Dynamic filter is a relative date filter, which filters data based on DynamicFilterType enumeration. Following code snippets illustrates how to apply Dynamic filter.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet sheet = workbook.Worksheets[0];
//Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
range.
sheet.AutoFilters.FilterRange = sheet.Range["A1:K180"];
//Column index to which AutoFilter must be applied.
IAutoFilter filter = sheet.AutoFilters[3];
//Applying dynamic filter to filter the date based on DynamicFilterType.
filter.AddDynamicFilter(DynamicFilterType.NextQuarter);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("Filter.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
```

```

'Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range.
sheet.AutoFilters.FilterRange = sheet.Range("A1:K180")
'Column index to which AutoFilter must be applied.
Dim filter As IAutoFilter = sheet.AutoFilters(3)
'Applying dynamic filter to filter the date based on DynamicFilterType.
filter.AddDynamicFilter(DynamicFilterType.NextQuarter)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Filter.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range.
    worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
    //Column index to which AutoFilter must be applied.
    IAutoFilter filter = worksheet.AutoFilters[3];
    //Applying dynamic filter to filter the date based on DynamicFilterType.
    filter.AddDynamicFilter(DynamicFilterType.NextQuarter);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Filter";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx" });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    workbook.Version = ExcelVersion.Excel2013;
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open, FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range.

```

```

worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
//Column index to which AutoFilter must be applied.
IAutoFilter filter = worksheet.AutoFilters[3];
//Applying dynamic filter to filter the date based on DynamicFilterType.
filter.AddDynamicFilter(DynamicFilterType.NextQuarter);
//Saving the workbook as stream
FileStream stream = new FileStream("Filter.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range.
    worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
    //Column index to which AutoFilter must be applied.
    IAutoFilter filter = worksheet.AutoFilters[3];
    //Applying dynamic filter to filter the date based on DynamicFilterType.
    filter.AddDynamicFilter(DynamicFilterType.NextQuarter);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer xlsio/xamarin section for respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Filter.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Filter.xlsx", "application/msexcel", stream);
    }
}

```


Color Filter

Color Filter can be used to filter data based on the color applied to the cell or the color applied to the text in the cell. The following code snippets show how to apply Color Filter based on Cell Color (fill color applied to the cell).

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range.
    sheet.AutoFilters.FilterRange = sheet.Range["A1:K180"];
    //Column index to which AutoFilter must be applied.
    IAutoFilter filter = sheet.AutoFilters[3];
    //Applying color filter to filter based on Cell Color.
    filter.AddColorFilter(Color.Red, ExcelColorFilterType.CellColor);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs("Filter.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range.
sheet.AutoFilters.FilterRange = sheet.Range("A1:K180")
'Column index to which AutoFilter must be applied.
Dim filter As IAutoFilter = sheet.AutoFilters(3)
'Applying color filter to filter based on Cell Color.
filter.AddColorFilter(Color.Red, ExcelColorFilterType.CellColor)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Filter.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range.
}
```

```

worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
//Column index to which AutoFilter must be applied.
IAutoFilter filter = worksheet.AutoFilters[3];
//Applying color filter to filter based on Cell Color.
filter.AddColorFilter(Color.FromArgb(255, 255, 0, 0),
ExcelColorFilterType.CellColor);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Filter";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
    range.
    worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
    //Column index to which AutoFilter must be applied.
    IAutoFilter filter = worksheet.AutoFilters[3];
    //Applying color filter to filter based on Cell Color.
    filter.AddColorFilter(Color.Red, ExcelColorFilterType.CellColor);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Filter.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    // "App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
    ple.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
}

```

```

//Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range.
worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
//Column index to which AutoFilter must be applied.
IAutoFilter filter = worksheet.AutoFilters[0];
//Applying color filter to filter based on Cell Color.
filter.AddColorFilter(Syncfusion.Drawing.Color.Red,
ExcelColorFilterType.CellColor);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Filter
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Filter.xlsx",
"application/msexcel", stream);
}
}

```

To filter cells based on Font color of the text inside cells just change the ExcelColorFilterType to Font Color. The following snippets show how to filter the cells based on font color.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet sheet = workbook.Worksheets[0];
//Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range.
sheet.AutoFilters.FilterRange = sheet.Range["A1:K180"];
//Column index to which AutoFilter must be applied.
IAutoFilter filter = sheet.AutoFilters[3];
//Applying color filter to filter based on Cell Color.
filter.AddColorFilter(Color.Red, ExcelColorFilterType.FontColor);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("Filter.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")

```

```

Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
range.
sheet.AutoFilters.FilterRange = sheet.Range("A1:K180")
'Column index to which AutoFilter must be applied.
Dim filter As IAutoFilter = sheet.AutoFilters(3)
'Applying color filter to filter based on Cell Color.
filter.AddColorFilter(Color.Red, ExcelColorFilterType.FontColor)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Filter.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
    range.
    worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
    //Column index to which AutoFilter must be applied.
    IAutoFilter filter = worksheet.AutoFilters[3];
    //Applying color filter to filter based on Cell Color.
    filter.AddColorFilter(Color.FromArgb(255, 255, 0, 0),
        ExcelColorFilterType.FontColor);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Filter";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    workbook.Version = ExcelVersion.Excel2013;
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
}

```

```
//Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
range.
worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
//Column index to which AutoFilter must be applied.
IAutoFilter filter = worksheet.AutoFilters[3];
//Applying color filter to filter based on Cell Color.
filter.AddColorFilter(Color.Red, ExcelColorFilterType.FontColor);
//Saving the workbook as stream
FileStream stream = new FileStream("Filter.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
range.
    worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
    //Column index to which AutoFilter must be applied.
    IAutoFilter filter = worksheet.AutoFilters[0];
    //Applying color filter to filter based on Cell Color.
    filter.AddColorFilter(Syncfusion.Drawing.Color.Red,
ExcelColorFilterType.FontColor);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Filter
.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Filter.xlsx",
"application/msexcel", stream);
    }
}
```

Icon Filter

Icon filter can be used to filter data that has conditional formatting with Icon Sets applied. Applying Icon Sets for numeric data adds icons to each cell based on the value present in that cell. Using Icon Filter, we can easily filter the data that has only a specific Icon in it.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
    range.
    sheet.AutoFilters.FilterRange = sheet.Range["A1:K180"];
    //Column index to which AutoFilter must be applied.
    IAutoFilter filter = sheet.AutoFilters[3];
    //Applying Icon filter to filter based on applied icon set.
    filter.AddIconFilter(ExcelIconSetType.ThreeFlags, 2);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs("Filter.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
range.
sheet.AutoFilters.FilterRange = sheet.Range("A1:K180")
'Column index to which AutoFilter must be applied.
Dim filter As IAutoFilter = sheet.AutoFilters(3)
'Applying Icon filter to filter based on applied icon set.
filter.AddIconFilter(ExcelIconSetType.ThreeFlags, 2)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Filter.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
}
```

```

//Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range.
worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
//Column index to which AutoFilter must be applied.
IAutoFilter filter = worksheet.AutoFilters[3];
//Applying Icon filter to filter based on applied icon set.
filter.AddIconFilter(ExcelIconSetType.ThreeFlags, 2);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Filter";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating an AutoFilter in the first worksheet. Specifying the AutoFilter range.
    worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
    //Column index to which AutoFilter must be applied.
    IAutoFilter filter = worksheet.AutoFilters[3];
    //Applying Icon filter to filter based on applied icon set.
    filter.AddIconFilter(ExcelIconSetType.ThreeFlags, 2);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Filter.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
}

```

```

//Creating an AutoFilter in the first worksheet. Specifying the AutoFilter
range.
worksheet.AutoFilters.FilterRange = worksheet.Range["A1:K180"];
//Column index to which AutoFilter must be applied.
IAutoFilter filter = worksheet.AutoFilters[3];
//Applying Icon filter to filter based on applied icon set.
filter.AddIconFilter(ExcelIconSetType.ThreeFlags, 2);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Filter
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Filter.xlsx",
"application/msexcel", stream);
}
}

```

Advanced Filter

Advanced Filter can be used to perform more complex filtering other than basic filters. You can filter a data with custom defined criteria range.

Advanced Filter support two types of filter action.

1. Filter In Place
2. Filter Copy

Filter In Place: Filter data in same location.

Filter Copy: Filter and copy data into new location within a worksheet.

Advanced Filter also provides an option to filter the unique records. This will remove the duplicate record from filtered data.

The following code illustrates how to apply Advanced Filter in worksheet.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("InputData.xlsx");
IWorksheet sheet = workbook.Worksheets[0];
IRange filterRange = sheet.Range["A1:C6"];
IRange criteriaRange = sheet.Range["A10:C12"];
}

```



```

IRange copyToRange = sheet.Range["K5:N5"];
//Apply the Advanced Filter with enable of unique value and copy to another
place.
sheet.AdvancedFilter(ExcelFilterAction.FilterCopy, filterRange,
criteriaRange, copyToRange, true);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("AdvancedFilter.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("InputData.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim filterRange As IRange = sheet.Range("A1:C6")
Dim criteriaRange As IRange = sheet.Range("A10:C12")
Dim copyToRange As IRange = sheet.Range("K5:N5")
'Apply the Advanced filter with enable of unique value and copy to another
place.
sheet.AdvancedFilter(ExcelFilterAction.FilterCopy, filterRange,
criteriaRange, copyToRange, True)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("AdvancedFilter.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
IRange filterRange = worksheet.Range["A1:C6"];
IRange criteriaRange = worksheet.Range["A10:C12"];
IRange copyToRange = worksheet.Range["K5:N5"];
//Apply the Advanced Filter with enable of unique value and copy to another
place.
worksheet.AdvancedFilter(ExcelFilterAction.FilterCopy, filterRange,
criteriaRange, copyToRange, true);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "AdvancedFilter";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file

```

```
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    IRange filterRange = worksheet.Range["A1:C6"];
    IRange criteriaRange = worksheet.Range["A10:C12"];
    IRange copyToRange = worksheet.Range["K5:N5"];
    //Apply the Advanced Filter with enable of unique value and copy to another
    place.
    worksheet.AdvancedFilter(ExcelFilterAction.FilterCopy, filterRange,
    criteriaRange, copyToRange, true);
    //Saving the workbook as stream
    FileStream stream = new FileStream("AdvancedFilter.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
    ple.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    IRange filterRange = worksheet.Range["A1:C6"];
    IRange criteriaRange = worksheet.Range["A10:C12"];
    IRange copyToRange = worksheet.Range["K5:N5"];
    //Apply the Advanced Filter with enable of unique value and copy to another
    place.
    worksheet.AdvancedFilter(ExcelFilterAction.FilterCopy, filterRange,
    criteriaRange, copyToRange, true);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
}
```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Filter
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Filter.xlsx",
"application/msexcel", stream);
}
}

```

Accessing Filter

We can access the filter and its criteria, based on its column index. Following code snippets illustrates how to apply access different types of filters using XlsIO.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//selecting the filter by column index
IAutoFilter filter = worksheet.AutoFilters[0];
switch (filter.FilterType)
{
case ExcelFilterType.CombinationFilter:
CombinationFilter filterItems = (filter.FilteredItems as CombinationFilter);
for (int index = 0; index < filterItems.Count; index++)
{
if (filterItems[index].CombinationFilterType ==
ExcelCombinationFilterType.TextFilter)
{
string textValue = (filterItems[index] as TextFilter).Text;
}
else
{
DateTimeGroupingType groupType = (filterItems[index] as
DateTimeFilter).GroupingType;
}
}
break;
case ExcelFilterType.DynamicFilter:
DynamicFilter dateFilter = (filter.FilteredItems as DynamicFilter);
DynamicFilterType dynamicFilterType = dateFilter.DateFilterType;
break;
case ExcelFilterType.CustomFilter:
IAutoFilterCondition firstCondition = filter.FirstCondition;
ExcelFilterDataType types = firstCondition.DataType;
break;
case ExcelFilterType.ColorFilter:
ColorFilter colorFilter = (filter.FilteredItems as ColorFilter);

```

```

Color color = colorFilter.Color;
ExcelColorFilterType filterType = colorFilter.ColorFilterType;
break;
case ExcelFilterType.IconFilter:
IconFilter iconFilter = (filter.FilteredItems as IconFilter);
int iconId = iconFilter.IconId;
ExcelIconSetType iconSetType = iconFilter.IconSetType;
break;
}
workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'selecting the filter by column index
Dim filter As IAutoFilter = worksheet.AutoFilters(0)
Select Case filter.FilterType
Case ExcelFilterType.ComboFilter
Dim filterItems As ComboFilter = TryCast(filter.FilteredItems,
ComboFilter)
For index As Integer = 0 To filterItems.Count - 1
If filterItems(index).ComboFilterType =
ExcelComboFilterType.TextFilter Then
Dim textValue As String = TryCast(filterItems(index), TextFilter).Text
Else
Dim groupType As DateTimeGroupingType = TryCast(filterItems(index),
DateTimeFilter).GroupingType
End If
Next
Exit Select
Case ExcelFilterType.DynamicFilter
Dim dateFilter As DynamicFilter = TryCast(filter.FilteredItems,
DynamicFilter)
Dim dynamicFilterType As DynamicFilterType = dateFilter.DateFilterType
Exit Select
Case ExcelFilterType.CustomFilter
Dim firstCondition As IAutoFilterCondition = filter.FirstCondition
Dim types As ExcelFilterDataType = firstCondition.DataType
Exit Select
Case ExcelFilterType.ColorFilter
Dim colorFilter As ColorFilter = TryCast(filter.FilteredItems, ColorFilter)
Dim color As Color = colorFilter.Color
Dim filterType As ExcelColorFilterType = colorFilter.ColorFilterType
Exit Select
Case ExcelFilterType.IconFilter
Dim iconFilter As IconFilter = TryCast(filter.FilteredItems, IconFilter)
Dim iconId As Int32 = iconFilter.IconId
Dim iconSetType As ExcelIconSetType = iconFilter.IconSetType
Exit Select
End Select

```

```
workbook.SaveAs ("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet sheet = workbook.Worksheets[0];
    //selecting the filter by column index
    IAutoFilter filter = sheet.AutoFilters[0];
    switch (filter.FilterType)
    {
        case ExcelFilterType.CombinationFilter:
            CombinationFilter filterItems = (filter.FilteredItems as CombinationFilter);
            for (int index = 0; index < filterItems.Count; index++)
            {
                if (filterItems[index].CombinationFilterType ==
                    ExcelCombinationFilterType.TextFilter)
                {
                    string textValue = (filterItems[index] as TextFilter).Text;
                }
                else
                {
                    DateTimeGroupingType groupType = (filterItems[index] as
                    DateTimeFilter).GroupingType;
                }
            }
            break;
        case ExcelFilterType.DynamicFilter:
            DynamicFilter dateFilter = (filter.FilteredItems as DynamicFilter);
            DynamicFilterType dynamicFilterType = dateFilter.DateFilterType;
            break;
        case ExcelFilterType.CustomFilter:
            IAutoFilterCondition firstCondition = filter.FirstCondition;
            ExcelFilterDataType types = firstCondition.DataType;
            break;
        case ExcelFilterType.ColorFilter:
            ColorFilter colorFilter = (filter.FilteredItems as ColorFilter);
            Color color = colorFilter.Color;
            ExcelColorFilterType filterType = colorFilter.ColorFilterType;
            break;
        case ExcelFilterType.IconFilter:
            IconFilter iconFilter = (filter.FilteredItems as IconFilter);
            int iconId = iconFilter.IconId;
            ExcelIconSetType iconSetType = iconFilter.IconSetType;
            break;
    }
}
```

```

}
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    //selecting the filter by column index
    IAutoFilter filter = sheet.AutoFilters[0];
    switch (filter.FilterType)
    {
        case ExcelFilterType.CombinationFilter:
            CombinationFilter filterItems = (filter.FilteredItems as CombinationFilter);
            for (int index = 0; index < filterItems.Count; index++)
            {
                if (filterItems[index].CombinationFilterType ==
                ExcelCombinationFilterType.TextFilter)
                {
                    string textValue = (filterItems[index] as TextFilter).Text;
                }
                else
                {
                    DateTimeGroupingType groupType = (filterItems[index] as
                    DateTimeFilter).GroupingType;
                }
            }
            break;
        case ExcelFilterType.DynamicFilter:
            DynamicFilter dateFilter = (filter.FilteredItems as DynamicFilter);
            DynamicFilterType dynamicFilterType = dateFilter.DateFilterType;
            break;
        case ExcelFilterType.CustomFilter:
            IAutoFilterCondition firstCondition = filter.FirstCondition;
            ExcelFilterDataType types = firstCondition.DataType;
            break;
        case ExcelFilterType.ColorFilter:
            ColorFilter colorFilter = (filter.FilteredItems as ColorFilter);
            Color color = colorFilter.Color;
            ExcelColorFilterType filterType = colorFilter.ColorFilterType;
            break;
    }
}

```

```

case ExcelFilterType.IconFilter:
    IconFilter iconFilter = (filter.FilteredItems as IconFilter);
    int iconId = iconFilter.IconId;
    ExcelIconSetType iconSetType = iconFilter.IconSetType;
    break;
}
//Saving the workbook as stream
FileStream file = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(file);
file.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    //selecting the filter by column index
    IAutoFilter filter = sheet.AutoFilters[0];
    switch (filter.FilterType)
    {
        case ExcelFilterType.ComboFilter:
            CombinationFilter filterItems = (filter.FilteredItems as CombinationFilter);
            for (int index = 0; index < filterItems.Count; index++)
            {
                if (filterItems[index].CombinationFilterType ==
                    ExcelCombinationFilterType.TextFilter)
                {
                    string textValue = (filterItems[index] as TextFilter).Text;
                }
                else
                {
                    DateTimeGroupingType groupType = (filterItems[index] as
                        DateTimeFilter).GroupingType;
                }
            }
            break;
        case ExcelFilterType.DynamicFilter:
            DynamicFilter dateFilter = (filter.FilteredItems as DynamicFilter);
            DynamicFilterType dynamicFilterType = dateFilter.DateFilterType;
            break;
        case ExcelFilterType.CustomFilter:
            IAutoFilterCondition firstCondition = filter.FirstCondition;
            ExcelFilterDataType types = firstCondition.DataType;
            break;
        case ExcelFilterType.ColorFilter:
            ColorFilter colorFilter = (filter.FilteredItems as ColorFilter);

```

```

Syncfusion.Drawing.Color color = colorFilter.Color;
ExcelColorFilterType filterType = colorFilter.ColorFilterType;
break;
case ExcelFilterType.IconFilter:
IconFilter iconFilter = (filter.FilteredItems as IconFilter);
int iconId = iconFilter.IconId;
ExcelIconSetType iconSetType = iconFilter.IconSetType;
break;
}
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Hyperlinks

You can create hyperlink in a workbook to provide quick access to web pages, places in your document and files. Hyperlink may target to any one of the following

- Worksheet range
- Web URL
- E-mail
- External files

The following code example illustrates how to insert various hyperlinks.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Creating a Hyperlink for a Website
IHyperLink hyperlink = sheet.HyperLinks.Add(sheet.Range["C5"]);
hyperlink.Type = ExcelHyperLinkType.Url;
hyperlink.Address = "http://www.syncfusion.com";
hyperlink.ScreenTip = "To know more about Syncfusion products, go through
this link.";
}

```



```

//Creating a Hyperlink for e-mail
IHyperLink hyperlink1 = sheet.HyperLinks.Add(sheet.Range["C7"]);
hyperlink1.Type = ExcelHyperLinkType.Url;
hyperlink1.Address = "mailto:Username@syncfusion.com";
hyperlink1.ScreenTip = "Send Mail";
//Creating a Hyperlink for Opening Files using type as File
IHyperLink hyperlink2 = sheet.HyperLinks.Add(sheet.Range["C9"]);
hyperlink2.Type = ExcelHyperLinkType.File;
hyperlink2.Address = @"C:\Program files";
hyperlink2.ScreenTip = "File path";
hyperlink2.TextToDisplay = "Hyperlink for files using File as type";
//Creating a Hyperlink for Opening Files using type as Unc
IHyperLink hyperlink3 = sheet.HyperLinks.Add(sheet.Range["C11"]);
hyperlink3.Type = ExcelHyperLinkType.Unc;
hyperlink3.Address = @"C:\Documents and Settings";
hyperlink3.ScreenTip = "Click here for files";
hyperlink3.TextToDisplay = "Hyperlink for files using Unc as type";
//Creating a Hyperlink to another cell using type as Workbook
IHyperLink hyperlink4 = sheet.HyperLinks.Add(sheet.Range["C13"]);
hyperlink4.Type = ExcelHyperLinkType.Workbook;
hyperlink4.Address = "Sheet1!A15";
hyperlink4.ScreenTip = "Click here";
hyperlink4.TextToDisplay = "Hyperlink to cell A15";
workbook.SaveAs("Hyperlink.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creating a Hyperlink for a Website
Dim hyperlink As IHyperLink = sheet.HyperLinks.Add(sheet.Range("C5"))
hyperlink.Type = ExcelHyperLinkType.Url
hyperlink.Address = "http://www.Syncfusion.com"
hyperlink.ScreenTip = "To know more about Syncfusion products, go through this link."
'Creating a Hyperlink for e-mail
Dim hyperlink1 As IHyperLink = sheet.HyperLinks.Add(sheet.Range("C7"))
hyperlink1.Type = ExcelHyperLinkType.Url
hyperlink1.Address = "mailto:Username@syncfusion.com"
hyperlink1.ScreenTip = "Send Mail"
'Creating a Hyperlink for Opening Files using type as File
Dim hyperlink2 As IHyperLink = sheet.HyperLinks.Add(sheet.Range("C9"))
hyperlink2.Type = ExcelHyperLinkType.File
hyperlink2.Address = "C:\Program files"
hyperlink2.ScreenTip = "File path"
hyperlink2.TextToDisplay = "Hyperlink for files using File as type"
'Creating a Hyperlink for Opening Files using type as Unc
Dim hyperlink3 As IHyperLink = sheet.HyperLinks.Add(sheet.Range("C11"))
hyperlink3.Type = ExcelHyperLinkType.Unc
hyperlink3.Address = "C:\Documents and Settings"
hyperlink3.ScreenTip = "Click here for files"
hyperlink3.TextToDisplay = "Hyperlink for files using Unc as type"

```

```
'Creating a Hyperlink to another cell using type as Workbook
Dim hyperlink4 As IHyperLink = sheet.HyperLinks.Add(sheet.Range("C13"))
hyperlink4.Type = ExcelHyperLinkType.Workbook
hyperlink4.Address = "Sheet1!A15"
hyperlink4.ScreenTip = "Click here"
hyperlink4.TextToDisplay = "Hyperlink to cell A15"
workbook.SaveAs("Hyperlink.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating a Hyperlink for a Website
    IHyperLink hyperlink = sheet.HyperLinks.Add(sheet.Range["C5"]);
    hyperlink.Type = ExcelHyperLinkType.Url;
    hyperlink.Address = "http://www.syncfusion.com";
    hyperlink.ScreenTip = "To know more about Syncfusion products, go through this link.";
    //Creating a Hyperlink for e-mail
    IHyperLink hyperlink1 = sheet.HyperLinks.Add(sheet.Range["C7"]);
    hyperlink1.Type = ExcelHyperLinkType.Url;
    hyperlink1.Address = "mailto:Username@syncfusion.com";
    hyperlink1.ScreenTip = "Send Mail";
    //Creating a Hyperlink for Opening Files using type as File
    IHyperLink hyperlink2 = sheet.HyperLinks.Add(sheet.Range["C9"]);
    hyperlink2.Type = ExcelHyperLinkType.File;
    hyperlink2.Address = @"C:\Program files";
    hyperlink2.ScreenTip = "File path";
    hyperlink2.TextToDisplay = "Hyperlink for files using File as type";
    //Creating a Hyperlink for Opening Files using type as Unc
    IHyperLink hyperlink3 = sheet.HyperLinks.Add(sheet.Range["C11"]);
    hyperlink3.Type = ExcelHyperLinkType.Unc;
    hyperlink3.Address = @"C:\Documents and Settings";
    hyperlink3.ScreenTip = "Click here for files";
    hyperlink3.TextToDisplay = "Hyperlink for files using Unc as type";
    //Creating a Hyperlink to another cell using type as Workbook
    IHyperLink hyperlink4 = sheet.HyperLinks.Add(sheet.Range["C13"]);
    hyperlink4.Type = ExcelHyperLinkType.Workbook;
    hyperlink4.Address = "Sheet1!A15";
    hyperlink4.ScreenTip = "Click here";
    hyperlink4.TextToDisplay = "Hyperlink to cell A15";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Hyperlink";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx" });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

```
}

```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating a Hyperlink for a Website
    IHyperLink hyperlink = sheet.HyperLinks.Add(sheet.Range["C5"]);
    hyperlink.Type = ExcelHyperLinkType.Url;
    hyperlink.Address = "http://www.syncfusion.com";
    hyperlink.ScreenTip = "To know more about Syncfusion products, go through this link.";
    //Creating a Hyperlink for e-mail
    IHyperLink hyperlink1 = sheet.HyperLinks.Add(sheet.Range["C7"]);
    hyperlink1.Type = ExcelHyperLinkType.Url;
    hyperlink1.Address = "mailto:Username@syncfusion.com";
    hyperlink1.ScreenTip = "Send Mail";
    //Creating a Hyperlink for Opening Files using type as File
    IHyperLink hyperlink2 = sheet.HyperLinks.Add(sheet.Range["C9"]);
    hyperlink2.Type = ExcelHyperLinkType.File;
    hyperlink2.Address = @"C:\Program files";
    hyperlink2.ScreenTip = "File path";
    hyperlink2.TextToDisplay = "Hyperlink for files using File as type";
    //Creating a Hyperlink for Opening Files using type as Unc
    IHyperLink hyperlink3 = sheet.HyperLinks.Add(sheet.Range["C11"]);
    hyperlink3.Type = ExcelHyperLinkType.Unc;
    hyperlink3.Address = @"C:\Documents and Settings";
    hyperlink3.ScreenTip = "Click here for files";
    hyperlink3.TextToDisplay = "Hyperlink for files using Unc as type";
    //Creating a Hyperlink to another cell using type as Workbook
    IHyperLink hyperlink4 = sheet.HyperLinks.Add(sheet.Range["C13"]);
    hyperlink4.Type = ExcelHyperLinkType.Workbook;
    hyperlink4.Address = "Sheet1!A15";
    hyperlink4.ScreenTip = "Click here";
    hyperlink4.TextToDisplay = "Hyperlink to cell A15";
    //Saving the workbook as stream
    FileStream stream = new FileStream("Hyperlink.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating a Hyperlink for a Website
    IHyperLink hyperlink = sheet.HyperLinks.Add(sheet.Range["C5"]);
}
```

```

hyperlink.Type = ExcelHyperLinkType.Url;
hyperlink.Address = "http://www.syncfusion.com";
hyperlink.ScreenTip = "To know more about Syncfusion products, go through
this link.";
//Creating a Hyperlink for e-mail
IHyperLink hyperlink1 = sheet.HyperLinks.Add(sheet.Range["C7"]);
hyperlink1.Type = ExcelHyperLinkType.Url;
hyperlink1.Address = "mailto:Username@syncfusion.com";
hyperlink1.ScreenTip = "Send Mail";
//Creating a Hyperlink for Opening Files using type as File
IHyperLink hyperlink2 = sheet.HyperLinks.Add(sheet.Range["C9"]);
hyperlink2.Type = ExcelHyperLinkType.File;
hyperlink2.Address = @"C:\Program files";
hyperlink2.ScreenTip = "File path";
hyperlink2.TextToDisplay = "Hyperlink for files using File as type";
//Creating a Hyperlink for Opening Files using type as Unc
IHyperLink hyperlink3 = sheet.HyperLinks.Add(sheet.Range["C11"]);
hyperlink3.Type = ExcelHyperLinkType.Unc;
hyperlink3.Address = @"C:\Documents and Settings";
hyperlink3.ScreenTip = "Click here for files";
hyperlink3.TextToDisplay = "Hyperlink for files using Unc as type";
//Creating a Hyperlink to another cell using type as Workbook
IHyperLink hyperlink4 = sheet.HyperLinks.Add(sheet.Range["C13"]);
hyperlink4.Type = ExcelHyperLinkType.Workbook;
hyperlink4.Address = "Sheet1!A15";
hyperlink4.ScreenTip = "Click here";
hyperlink4.TextToDisplay = "Hyperlink to cell A15";
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Hyperl
ink.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Hyperlink.xlsx",
"application/msexcel", stream);
}
}

```

Modifying Existing Hyperlink

You can modify the properties of existing hyperlink by accessing the Hyperlinks collection of the IRange instance.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{

```

```

IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet sheet = workbook.Worksheets[0];
//Modifying hyperlink's text to display
IHyperLink hyperlink = sheet.Range["C5"].Hyperlinks[0];
hyperlink.TextToDisplay = "Syncfusion";
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("Hyperlink.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Modifying hyperlink's text to display
Dim hyperlink As IHyperLink = sheet.Range("C5").Hyperlinks(0)
hyperlink.TextToDisplay = "Syncfusion"
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Hyperlink.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Modifying hyperlink's text to display
IHyperLink hyperlink = worksheet.Range["C5"].Hyperlinks[0];
hyperlink.TextToDisplay = "Syncfusion";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Hyperlink";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Modifying hyperlink's text to display
    IHyperLink hyperlink = worksheet.Range["C5"].Hyperlinks[0];
    hyperlink.TextToDisplay = "Syncfusion";
    //Saving the workbook as stream
    FileStream stream = new FileStream("Hyperlink.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
    ple.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Modifying hyperlink's text to display
    IHyperLink hyperlink = worksheet.Range["C5"].Hyperlinks[0];
    hyperlink.TextToDisplay = "Syncfusion";
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Hyperl
        ink.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Hyperlink.xlsx",
        "application/msexcel", stream);
    }
}

```

Removing Hyperlink

You can remove a hyperlink from a range by accessing the Hyperlinks collection of the IRange instance.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Removing Hyperlink from Range "C7"
    sheet.Range["C7"].Hyperlinks.RemoveAt(0);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs("Hyperlink.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Removing Hyperlink from Range "C7"
sheet.Range("C7").Hyperlinks.RemoveAt(0)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Hyperlink.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Removing Hyperlink from Range "C7"
    worksheet.Range["C7"].Hyperlinks.RemoveAt(0);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Hyperlink";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx" });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    workbook.Version = ExcelVersion.Excel2013;
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Removing Hyperlink from Range "C7"
    worksheet.Range["C7"].Hyperlinks.RemoveAt(0);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Hyperlink.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Removing Hyperlink from Range "C7"
    worksheet.Range["C7"].Hyperlinks.RemoveAt(0);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Hyperlink.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Hyperlink.xlsx", "application/msexcel", stream);
    }
}

```


*Hyperlinks on Shapes***Adding Hyperlinks to Shapes**

Hyperlink can be added to the following shapes.

- Picture
- AutoShape
- TextBox

The following code example illustrates how to insert hyperlinks to shapes.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Adding hyperlink to TextBox
    IWorksheet sheet = workbook.Worksheets[0];
    ITextBox textBox = sheet.TextBoxes.AddTextBox(1, 1, 100, 100);
    IHyperLink hyperlink = sheet.HyperLinks.Add((textBox as IShape),
    ExcelHyperLinkType.Url, "http://www.Syncfusion.com", "click here");
    //Adding hyperlink to AutoShape
    sheet = workbook.Worksheets[1];
    IShape autoShape = sheet.Shapes.AddAutoShapes(AutoShapeType.Cloud, 1, 1,
    100, 100);
    hyperlink = sheet.HyperLinks.Add(autoShape, ExcelHyperLinkType.Url,
    "mailto:Username@syncfusion.com", "Send Mail");
    //Adding hyperlink to picture
    sheet = workbook.Worksheets[2];
    IPictureShape picture = sheet.Pictures.AddPicture(@"Image.png");
    hyperlink = sheet.HyperLinks.Add(picture);
    hyperlink.Type = ExcelHyperLinkType.Unc;
    hyperlink.Address = "C:\\Documents and Settings";
    hyperlink.ScreenTip = "Click here for files";
    workbook.SaveAs("Hyperlink.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Text box
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim textBox As ITextBox = sheet.TextBoxes.AddTextBox(1, 1, 100, 100)
Dim hyperlink As IHyperLink = sheet.HyperLinks.Add(TryCast(textBox, IShape),
ExcelHyperLinkType.Url, "http://www.Syncfusion.com", "click here")
'AutoShapes
sheet = workbook.Worksheets(1)
Dim autoShape As IShape = sheet.Shapes.AddAutoShapes(AutoShapeType.Cloud, 1,
1, 100, 100)
```

```

hyperlink = sheet.HyperLinks.Add(autoShape, ExcelHyperLinkType.Url,
"mailto:Username@syncfusion.com", "Send Mail")
'Pictures
sheet = workbook.Worksheets(2)
Dim picture As IPictureShape = sheet.Pictures.AddPicture("Image.png")
hyperlink = sheet.HyperLinks.Add(picture)
hyperlink.Type = ExcelHyperLinkType.Unc
hyperlink.Address = "C:\Documents and Settings"
hyperlink.ScreenTip = "Click here for files"
workbook.SaveAs("Hyperlink.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding hyperlink to TextBox
    ITextBox textBox = worksheet.TextBoxes.AddTextBox(1, 1, 100, 100);
    IHyperLink hyperlink = worksheet.HyperLinks.Add((textBox as IShape),
ExcelHyperLinkType.Url, "http://www.Syncfusion.com", "click here");
    //Adding hyperlink to AutoShape
    IShape autoShape = worksheet.Shapes.AddAutoShapes(AutoShapeType.Cloud, 1, 1,
100, 100);
    hyperlink = worksheet.HyperLinks.Add(autoShape, ExcelHyperLinkType.Url,
"mailto:Username@syncfusion.com", "Send Mail");
    //Adding hyperlink to picture
    IPictureShape picture = worksheet.Pictures.AddPictureAsLink(5, 5, 10, 10,
"Image.png");
    hyperlink = worksheet.HyperLinks.Add(picture);
    hyperlink.Type = ExcelHyperLinkType.Unc;
    hyperlink.Address = "C:\\Documents and Settings";
    hyperlink.ScreenTip = "Click here for files";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Hyperlink";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    workbook.Version = ExcelVersion.Excel2013;
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding hyperlink to TextBox
    ITextBox textBox = worksheet.TextBoxes.AddTextBox(1, 1, 100, 100);
    IHyperLink hyperlink = worksheet.HyperLinks.Add((textBox as IShape),
    ExcelHyperLinkType.Url, "http://www.Syncfusion.com", "click here");
    //Adding hyperlink to AutoShape
    IShape autoShape = worksheet.Shapes.AddAutoShapes(AutoShapeType.Cloud, 1, 1,
    100, 100);
    hyperlink = worksheet.HyperLinks.Add(autoShape, ExcelHyperLinkType.Url,
    "mailto:Username@syncfusion.com", "Send Mail");
    //Adding hyperlink to picture
    IPictureShape picture = worksheet.Pictures.AddPictureAsLink(5, 5, 10, 10,
    "Image.png");
    hyperlink = worksheet.HyperLinks.Add(picture);
    hyperlink.Type = ExcelHyperLinkType.Unc;
    hyperlink.Address = "C:\\Documents and Settings";
    hyperlink.ScreenTip = "Click here for files";
    //Saving the workbook as stream
    FileStream stream = new FileStream("Hyperlink.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
    ple.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding hyperlink to TextBox
    ITextBox textBox = worksheet.TextBoxes.AddTextBox(1, 1, 100, 100);
    IHyperLink hyperlink = worksheet.HyperLinks.Add((textBox as IShape),
    ExcelHyperLinkType.Url, "http://www.Syncfusion.com", "click here");
    //Adding hyperlink to AutoShape
    IShape autoShape = worksheet.Shapes.AddAutoShapes(AutoShapeType.Cloud, 1, 1,
    100, 100);
    hyperlink = worksheet.HyperLinks.Add(autoShape, ExcelHyperLinkType.Url,
    "mailto:Username@syncfusion.com", "Send Mail");
    //Adding hyperlink to picture

```

```

IPictureShape picture = worksheet.Pictures.AddPictureAsLink(5, 5, 10, 10,
"Image.png");
hyperlink = worksheet.HyperLinks.Add(picture);
hyperlink.Type = ExcelHyperLinkType.Unc;
hyperlink.Address = "C:\\Documents and Settings";
hyperlink.ScreenTip = "Click here for files";
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Hyperl
ink.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Hyperlink.xlsx",
"application/msexcel", stream);
}
}

```

Modifying Hyperlinks on Shapes

Properties of existing hyperlink can be modified by accessing either the Hyperlinks collection of the IWorksheet instance or Hyperlink property in respective IShape instance.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet sheet = workbook.Worksheets[0];
//Modifying hyperlink's screen tip through IWorksheet instance
IHyperLink hyperlink = sheet.HyperLinks[0];
hyperlink.ScreenTip = "Syncfusion";
//Modifying hyperlink's screen tip through IShape instance
hyperlink = sheet.Shapes[0].Hyperlink;
hyperlink.ScreenTip = "Syncfusion";
workbook.SaveAs("Hyperlink.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)

```

```

'Modifying hyperlink's screen tip through IWorksheet instance
Dim hyperlink As IHyperLink = sheet.HyperLinks(0)
hyperlink.ScreenTip = "Syncfusion"
'Modifying hyperlink's screen tip through IShape instance
hyperlink = sheet.Shapes(0).Hyperlink
hyperlink.ScreenTip = "Syncfusion"
workbook.SaveAs("Hyperlink.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Modifying hyperlink's screen tip through IWorksheet instance
    IHyperLink hyperlink = worksheet.HyperLinks[0];
    hyperlink.ScreenTip = "Syncfusion";
    //Modifying hyperlink's screen tip through IShape instance
    hyperlink = worksheet.Shapes[0].Hyperlink;
    hyperlink.ScreenTip = "Syncfusion";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Hyperlink";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Modifying hyperlink's screen tip through IWorksheet instance
    IHyperLink hyperlink = worksheet.HyperLinks[0];
    hyperlink.ScreenTip = "Syncfusion";
    //Modifying hyperlink's screen tip through IShape instance

```

```

hyperlink = worksheet.Shapes[0].Hyperlink;
hyperlink.ScreenTip = "Syncfusion";
//Saving the workbook as stream
FileStream stream = new FileStream("Hyperlink.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Modifying hyperlink's screen tip through IWorksheet instance
    IHyperLink hyperlink = worksheet.HyperLinks[0];
    hyperlink.ScreenTip = "Syncfusion";
    //Modifying hyperlink's screen tip through IShape instance
    hyperlink = worksheet.Shapes[0].Hyperlink;
    hyperlink.ScreenTip = "Syncfusion";
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Hyperlink.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Hyperlink.xlsx", "application/msexcel", stream);
    }
}

```

Removing Hyperlinks from Shapes

Hyperlinks from shapes can be removed by accessing Hyperlinks collection of the worksheet instance.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())

```

```
{
  IApplication application = excelEngine.Excel;
  application.DefaultVersion = ExcelVersion.Excel2013;
  IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
  IWorksheet sheet = workbook.Worksheets[0];
  //Removing hyperlink from sheet with Index
  sheet.HyperLinks.RemoveAt(0);
  workbook.SaveAs("Hyperlink.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Removing Hyperlink from sheet with Index
sheet.HyperLinks.RemoveAt(0)
workbook.SaveAs("Hyperlink.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
  IApplication application = excelEngine.Excel;
  application.DefaultVersion = ExcelVersion.Excel2013;
  //Instantiates the File Picker
  FileOpenPicker openPicker = new FileOpenPicker();
  openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
  openPicker.FileTypeFilter.Add(".xlsx");
  openPicker.FileTypeFilter.Add(".xls");
  StorageFile file = await openPicker.PickSingleFileAsync();
  //Opens the workbook
  IWorkbook workbook = await application.Workbooks.OpenAsync(file);
  IWorksheet worksheet = workbook.Worksheets[0];
  //Removing hyperlink from sheet with Index
  worksheet.HyperLinks.RemoveAt(0);
  //Initializes FileSavePicker
  FileSavePicker savePicker = new FileSavePicker();
  savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
  savePicker.SuggestedFileName = "Hyperlink";
  savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
  //Creates a storage file from FileSavePicker
  StorageFile storageFile = await savePicker.PickSaveFileAsync();
  //Saves changes to the specified storage file
  await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
```

```

IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream);
IWorksheet worksheet = workbook.Worksheets[0];
//Removing hyperlink from sheet with Index
worksheet.HyperLinks.RemoveAt(0);
//Saving the workbook as stream
FileStream stream = new FileStream("Hyperlink.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Removing hyperlink from sheet with Index
    worksheet.HyperLinks.RemoveAt(0);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Hyperlink.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Hyperlink.xlsx", "application/msexcel", stream);
    }
}

```

Working with Cell or Range Formatting

This section covers the various formatting options in a cell or a range.

Create a Style

The following code shows how to create and apply cell style.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating a new style with cell back color, fill pattern and font attribute
    IStyle style = workbook.Styles.Add("NewStyle");
    style.Color = Color.LightGreen;
    style.FillPattern = ExcelPattern.DarkUpwardDiagonal;
    style.Font.Bold = true;
    worksheet.Range["B2"].CellStyle = style;
    workbook.SaveAs("Style.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Creating a new style with cell back color, fill pattern and font attribute
Dim style As IStyle = workbook.Styles.Add("NewStyle")
style.Color = Color.LightGreen
style.FillPattern = ExcelPattern.DarkUpwardDiagonal
style.Font.Bold = True
worksheet.Range("B2").CellStyle = style
workbook.SaveAs("Style.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating a new style with cell back color, fill pattern and font attribute
    IStyle style = workbook.Styles.Add("NewStyle");
    style.Color = Color.FromArgb(255, 144, 238, 144);
    style.FillPattern = ExcelPattern.DarkUpwardDiagonal;
    style.Font.Bold = true;
    worksheet.Range["B2"].CellStyle = style;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Style";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
```

```
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating a new style with cell back color, fill pattern and font attribute
    IStyle style = workbook.Styles.Add("NewStyle");
    style.Color = Color.LightGreen;
    style.FillPattern = ExcelPattern.DarkUpwardDiagonal;
    style.Font.Bold = true;
    worksheet.Range["B2"].CellStyle = style;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Style.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating a new style with cell back color, fill pattern and font attribute
    IStyle style = workbook.Styles.Add("NewStyle");
    style.Color = Syncfusion.Drawing.Color.LightGreen;
    style.FillPattern = ExcelPattern.DarkUpwardDiagonal;
    style.Font.Bold = true;
    worksheet.Range["B2"].CellStyle = style;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Style.
        xlsx", "application/msexcel", stream);
    }
    else
```

```
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Style.xlsx",
    "application/msexcel", stream);
}
```

Set Default Style for row or column

It is the recommended and optimized approach to format entire row or column with same styles instead of formatting each and every cell individually. Use the following code to set default style.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Define new styles to apply in rows and columns
    IStyle rowStyle = workbook.Styles.Add("RowStyle");
    rowStyle.Color = Color.LightGreen;
    IStyle columnStyle = workbook.Styles.Add("ColumnStyle");
    columnStyle.Color = Color.Orange;
    //Set default row style for entire row
    worksheet.SetDefaultRowStyle(1, 2, rowStyle);
    //Set default column style for entire column
    worksheet.SetDefaultColumnStyle(1, 2, columnStyle);
    workbook.SaveAs("DefaultStyles.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Define new styles to apply in rows and columns
Dim rowStyle As IStyle = workbook.Styles.Add("RowStyle")
rowStyle.Color = Color.LightGreen
Dim columnStyle As IStyle = workbook.Styles.Add("ColumnStyle")
columnStyle.Color = Color.Orange
'Set default row style for entire row
worksheet.SetDefaultRowStyle(1, 2, rowStyle)
'Set default column style for entire column
worksheet.SetDefaultColumnStyle(1, 2, columnStyle)
workbook.SaveAs("DefaultStyles.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
```

```

IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Define new styles to apply in rows and columns
IStyle rowStyle = workbook.Styles.Add("RowStyle");
rowStyle.Color = Color.FromArgb(255, 144, 238, 144);
IStyle columnStyle = workbook.Styles.Add("ColumnStyle");
columnStyle.Color = Color.FromArgb(255, 255, 165, 0);
//Set default row style for entire row
worksheet.SetDefaultRowStyle(1, 2, rowStyle);
//Set default column style for entire column
worksheet.SetDefaultColumnStyle(1, 2, columnStyle);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "DefaultStyles";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Define new styles to apply in rows and columns
    IStyle rowStyle = workbook.Styles.Add("RowStyle");
    rowStyle.Color = Color.LightGreen;
    IStyle columnStyle = workbook.Styles.Add("ColumnStyle");
    columnStyle.Color = Color.Orange;
    //Set default row style for entire row
    worksheet.SetDefaultRowStyle(1, 2, rowStyle);
    //Set default column style for entire column
    worksheet.SetDefaultColumnStyle(1, 2, columnStyle);
    //Saving the workbook as stream
    FileStream stream = new FileStream("DefaultStyles.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Define new styles to apply in rows and columns

```

```

IStyle rowStyle = workbook.Styles.Add("RowStyle");
rowStyle.Color = Syncfusion.Drawing.Color.LightGreen;
IStyle columnStyle = workbook.Styles.Add("ColumnStyle");
columnStyle.Color = Syncfusion.Drawing.Color.Orange;
//Set default row style for entire row
worksheet.SetDefaultRowStyle(1, 2, rowStyle);
//Set default column style for entire column
worksheet.SetDefaultColumnStyle(1, 2, columnStyle);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android, and iOS platforms. Refer to the xlsio/xamarin section for
//respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Default
tStyles.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("DefaultStyles.xlsx
", "application/msexcel", stream);
}
}

```

Note: Applying custom styles will override original styles.

Tips: To apply styles for whole column instead of applying in each cell, use default styles.

Apply Global Style

The XlsIO adds styles globally that can be applied to one or more cells in a workbook. This is a recommended approach to apply single style in different rows and columns, which improves memory and performance considerably.

To learn more about performance, refer to the [Improving Performing section for better performance in XlsIO](#).

The following code snippet illustrates how to set header style and body style to the cells.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(2);
IWorksheet worksheet = workbook.Worksheets[0];
//Adding values to a worksheet range
worksheet.Range["A1"].Text = "CustomerID";
worksheet.Range["B1"].Text = "CompanyName";
worksheet.Range["C1"].Text = "ContactName";
worksheet.Range["A2"].Text = "ALFKI";
worksheet.Range["A3"].Text = "ANATR";
}

```

```

worksheet.Range["A4"].Text = "BONAP";
worksheet.Range["A5"].Text = "BSBEV";
worksheet.Range["B2"].Text = "Alfred Futterkiste";
worksheet.Range["B3"].Text = "Ana Trujillo Emparedados y helados";
worksheet.Range["B4"].Text = "Bon App";
worksheet.Range["B5"].Text = "B's Beverages";
worksheet.Range["C2"].Text = "Maria Anders";
worksheet.Range["C3"].Text = "Ana Trujillo";
worksheet.Range["C4"].Text = "Laurence Lebihan";
worksheet.Range["C5"].Text = "Victoria Ashworth";
//Formatting
//Global styles should be used when the same style needs to be applied to
more than one cell. This usage of a global style reduces memory usage.
//Add custom colors to the palette
workbook.SetPaletteColor(8, Color.FromArgb(255, 174, 33));
//Defining header style
IStyle headerStyle = workbook.Styles.Add("HeaderStyle");
headerStyle.BeginUpdate();
headerStyle.Color = Color.FromArgb(255, 174, 33);
headerStyle.Font.Bold = true;
headerStyle.Borders[ExcelBordersIndex.EdgeLeft].LineStyle =
ExcelLineStyle.Thin;
headerStyle.Borders[ExcelBordersIndex.EdgeRight].LineStyle =
ExcelLineStyle.Thin;
headerStyle.Borders[ExcelBordersIndex.EdgeTop].LineStyle =
ExcelLineStyle.Thin;
headerStyle.Borders[ExcelBordersIndex.EdgeBottom].LineStyle =
ExcelLineStyle.Thin;
headerStyle.EndUpdate();
//Add custom colors to the palette
workbook.SetPaletteColor(9, Color.FromArgb(239, 243, 247));
//Defining body style
IStyle bodyStyle = workbook.Styles.Add("BodyStyle");
bodyStyle.BeginUpdate();
bodyStyle.Color = Color.FromArgb(239, 243, 247);
bodyStyle.Borders[ExcelBordersIndex.EdgeLeft].LineStyle =
ExcelLineStyle.Thin;
bodyStyle.Borders[ExcelBordersIndex.EdgeRight].LineStyle =
ExcelLineStyle.Thin;
bodyStyle.EndUpdate();
//Apply Header style
worksheet.Rows[0].CellStyle = headerStyle;
//Apply Body Style
worksheet.Range["A2:C5"].CellStyle = bodyStyle;
//Auto-fit the columns
worksheet.UsedRange.AutofitColumns();
workbook.SaveAs("GlobalStyles.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(2)
Dim worksheet As IWorksheet = workbook.Worksheets(0)

```

```

'Adding values to a worksheet range
worksheet.Range("A1").Text = "CustomerID"
worksheet.Range("B1").Text = "CompanyName"
worksheet.Range("C1").Text = "ContactName"
worksheet.Range("A2").Text = "ALFKI"
worksheet.Range("A3").Text = "ANATR"
worksheet.Range("A4").Text = "BONAP"
worksheet.Range("A5").Text = "BSBEV"
worksheet.Range("B2").Text = "Alfred Futterkiste"
worksheet.Range("B3").Text = "Ana Trujillo Emparedados y helados"
worksheet.Range("B4").Text = "Bon App"
worksheet.Range("B5").Text = "B's Beverages"
worksheet.Range("C2").Text = "Maria Anders"
worksheet.Range("C3").Text = "Ana Trujillo"
worksheet.Range("C4").Text = "Laurence Lebihan"
worksheet.Range("C5").Text = "Victoria Ashworth"

'Formatting
'Global styles should be used when the same style needs to be applied to
more than one cell. This usage of a global style reduces memory usage.
'Add custom colors to the palette
workbook.SetPaletteColor(8, Color.FromArgb(255, 174, 33))
'Defining header style
Dim headerStyle As IStyle = workbook.Styles.Add("HeaderStyle")
headerStyle.BeginUpdate()
headerStyle.Color = Color.FromArgb(255, 174, 33)
headerStyle.Font.Bold = True
headerStyle.Borders(ExcelBordersIndex.EdgeLeft).LineStyle =
ExcelLineStyle.Thin
headerStyle.Borders(ExcelBordersIndex.EdgeRight).LineStyle =
ExcelLineStyle.Thin
headerStyle.Borders(ExcelBordersIndex.EdgeTop).LineStyle =
ExcelLineStyle.Thin
headerStyle.Borders(ExcelBordersIndex.EdgeBottom).LineStyle =
ExcelLineStyle.Thin
headerStyle.EndUpdate()
'Add custom colors to the palette
workbook.SetPaletteColor(9, Color.FromArgb(239, 243, 247))
'Defining body style
Dim bodyStyle As IStyle = workbook.Styles.Add("BodyStyle")
bodyStyle.BeginUpdate()
bodyStyle.Color = Color.FromArgb(239, 243, 247)
bodyStyle.Borders(ExcelBordersIndex.EdgeLeft).LineStyle =
ExcelLineStyle.Thin
bodyStyle.Borders(ExcelBordersIndex.EdgeRight).LineStyle =
ExcelLineStyle.Thin
bodyStyle.EndUpdate()
'Apply Header style
worksheet.Rows(0).CellStyle = headerStyle
'Apply Body Style
worksheet.Range("A2:C5").CellStyle = bodyStyle
'Auto-fit the columns
worksheet.UsedRange.AutofitColumns()
workbook.SaveAs("GlobalStyles.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding values to a worksheet range
    worksheet.Range["A1"].Text = "CustomerID";
    worksheet.Range["B1"].Text = "CompanyName";
    worksheet.Range["C1"].Text = "ContactName";
    worksheet.Range["A2"].Text = "ALFKI";
    worksheet.Range["A3"].Text = "ANATR";
    worksheet.Range["A4"].Text = "BONAP";
    worksheet.Range["A5"].Text = "BSBEV";
    worksheet.Range["B2"].Text = "Alfred Futterkiste";
    worksheet.Range["B3"].Text = "Ana Trujillo Emparedados y helados";
    worksheet.Range["B4"].Text = "Bon App";
    worksheet.Range["B5"].Text = "B's Beverages";
    worksheet.Range["C2"].Text = "Maria Anders";
    worksheet.Range["C3"].Text = "Ana Trujillo";
    worksheet.Range["C4"].Text = "Laurence Lebihan";
    worksheet.Range["C5"].Text = "Victoria Ashworth";
    //Formatting
    //Global styles should be used when the same style needs to be applied to
more than one cell. This usage of a global style reduces memory usage.
    //Add custom colors to the palette
    workbook.SetPaletteColor(8, Color.FromArgb(255, 255, 174, 33));
    //Defining header style
    IStyle headerStyle = workbook.Styles.Add("HeaderStyle");
    headerStyle.BeginUpdate();
    headerStyle.Color = Color.FromArgb(255, 255, 174, 33);
    headerStyle.Font.Bold = true;
    headerStyle.Borders[ExcelBordersIndex.EdgeLeft].LineStyle =
    ExcelLineStyle.Thin;
    headerStyle.Borders[ExcelBordersIndex.EdgeRight].LineStyle =
    ExcelLineStyle.Thin;
    headerStyle.Borders[ExcelBordersIndex.EdgeTop].LineStyle =
    ExcelLineStyle.Thin;
    headerStyle.Borders[ExcelBordersIndex.EdgeBottom].LineStyle =
    ExcelLineStyle.Thin;
    headerStyle.EndUpdate();
    //Add custom colors to the palette
    workbook.SetPaletteColor(9, Color.FromArgb(255, 239, 243, 247));
    //Defining body style
    IStyle bodyStyle = workbook.Styles.Add("BodyStyle");
    bodyStyle.BeginUpdate();
    bodyStyle.Color = Color.FromArgb(255, 239, 243, 247);
    bodyStyle.Borders[ExcelBordersIndex.EdgeLeft].LineStyle =
    ExcelLineStyle.Thin;
    bodyStyle.Borders[ExcelBordersIndex.EdgeRight].LineStyle =
    ExcelLineStyle.Thin;
    bodyStyle.EndUpdate();
    //Apply Header style
    worksheet.Rows[0].CellStyle = headerStyle;
    //Apply Body Style
    worksheet.Range["A2:C5"].CellStyle = bodyStyle;
    //Auto-fit the columns

```



```

worksheet.UsedRange.AutofitColumns();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "GlobalStyles";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding values to a worksheet range
    worksheet.Range["A1"].Text = "CustomerID";
    worksheet.Range["B1"].Text = "CompanyName";
    worksheet.Range["C1"].Text = "ContactName";
    worksheet.Range["A2"].Text = "ALFKI";
    worksheet.Range["A3"].Text = "ANATR";
    worksheet.Range["A4"].Text = "BONAP";
    worksheet.Range["A5"].Text = "BSBEV";
    worksheet.Range["B2"].Text = "Alfred Futterkiste";
    worksheet.Range["B3"].Text = "Ana Trujillo Emparedados y helados";
    worksheet.Range["B4"].Text = "Bon App";
    worksheet.Range["B5"].Text = "B's Beverages";
    worksheet.Range["C2"].Text = "Maria Anders";
    worksheet.Range["C3"].Text = "Ana Trujillo";
    worksheet.Range["C4"].Text = "Laurence Lebihan";
    worksheet.Range["C5"].Text = "Victoria Ashworth";
    //Formatting
    //Global styles should be used when the same style needs to be applied to
    more than one cell. This usage of a global style reduces memory usage.
    //Add custom colors to the palette
    workbook.SetPaletteColor(8, Color.FromArgb(255, 174, 33));
    //Defining header style
    IStyle headerStyle = workbook.Styles.Add("HeaderStyle");
    headerStyle.BeginUpdate();
    headerStyle.Color = Color.FromArgb(255, 174, 33);
    headerStyle.Font.Bold = true;
    headerStyle.Borders[ExcelBordersIndex.EdgeLeft].LineStyle =
    ExcelLineStyle.Thin;
    headerStyle.Borders[ExcelBordersIndex.EdgeRight].LineStyle =
    ExcelLineStyle.Thin;
    headerStyle.Borders[ExcelBordersIndex.EdgeTop].LineStyle =
    ExcelLineStyle.Thin;
    headerStyle.Borders[ExcelBordersIndex.EdgeBottom].LineStyle =
    ExcelLineStyle.Thin;
    headerStyle.EndUpdate();
}

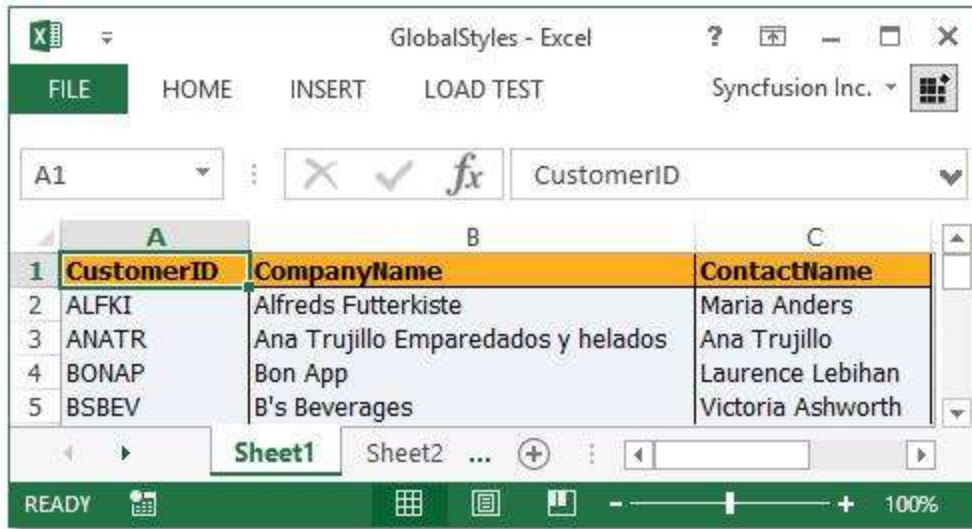
```

```
//Add custom colors to the palette
workbook.SetPaletteColor(9, Color.FromArgb(239, 243, 247));
//Defining body style
IStyle bodyStyle = workbook.Styles.Add("BodyStyle");
bodyStyle.BeginUpdate();
bodyStyle.Color = Color.FromArgb(239, 243, 247);
bodyStyle.Borders[ExcelBordersIndex.EdgeLeft].LineStyle =
ExcelLineStyle.Thin;
bodyStyle.Borders[ExcelBordersIndex.EdgeRight].LineStyle =
ExcelLineStyle.Thin;
bodyStyle.EndUpdate();
//Apply Header style
worksheet.Rows[0].CellStyle = headerStyle;
//Apply Body Style
worksheet.Range["A2:C5"].CellStyle = bodyStyle;
//Auto-fit the columns
worksheet.UsedRange.AutofitColumns();
//Saving the workbook as stream
FileStream stream = new FileStream("GlobalStyles.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding values to a worksheet range
    worksheet.Range["A1"].Text = "CustomerID";
    worksheet.Range["B1"].Text = "CompanyName";
    worksheet.Range["C1"].Text = "ContactName";
    worksheet.Range["A2"].Text = "ALFKI";
    worksheet.Range["A3"].Text = "ANATR";
    worksheet.Range["A4"].Text = "BONAP";
    worksheet.Range["A5"].Text = "BSBEV";
    worksheet.Range["B2"].Text = "Alfred Futterkiste";
    worksheet.Range["B3"].Text = "Ana Trujillo Emparedados y helados";
    worksheet.Range["B4"].Text = "Bon App";
    worksheet.Range["B5"].Text = "B's Beverages";
    worksheet.Range["C2"].Text = "Maria Anders";
    worksheet.Range["C3"].Text = "Ana Trujillo";
    worksheet.Range["C4"].Text = "Laurence Lebihan";
    worksheet.Range["C5"].Text = "Victoria Ashworth";
    //Formatting
    //Global styles should be used when the same style needs to be applied to
    more than one cell. This usage of a global style reduces memory usage.
    //Add custom colors to the palette
    workbook.SetPaletteColor(8, Syncfusion.Drawing.Color.FromArgb(255, 174,
    33));
    //Defining header style
    IStyle headerStyle = workbook.Styles.Add("HeaderStyle");
}
```

```
headerStyle.BeginUpdate();
headerStyle.Color = Syncfusion.Drawing.Color.FromArgb(255, 174, 33);
headerStyle.Font.Bold = true;
headerStyle.Borders[ExcelBordersIndex.EdgeLeft].LineStyle =
ExcelLineStyle.Thin;
headerStyle.Borders[ExcelBordersIndex.EdgeRight].LineStyle =
ExcelLineStyle.Thin;
headerStyle.Borders[ExcelBordersIndex.EdgeTop].LineStyle =
ExcelLineStyle.Thin;
headerStyle.Borders[ExcelBordersIndex.EdgeBottom].LineStyle =
ExcelLineStyle.Thin;
headerStyle.EndUpdate();
//Add custom colors to the palette
workbook.SetPaletteColor(9, Syncfusion.Drawing.Color.FromArgb(255, 239, 243,
247));
//Defining body style
IStyle bodyStyle = workbook.Styles.Add("BodyStyle");
bodyStyle.BeginUpdate();
bodyStyle.Color = Syncfusion.Drawing.Color.FromArgb(255, 239, 243, 247);
bodyStyle.Borders[ExcelBordersIndex.EdgeLeft].LineStyle =
ExcelLineStyle.Thin;
bodyStyle.Borders[ExcelBordersIndex.EdgeRight].LineStyle =
ExcelLineStyle.Thin;
bodyStyle.EndUpdate();
//Apply Header style
worksheet.Rows[0].CellStyle = headerStyle;
//Apply Body Style
worksheet.Range["A2:C5"].CellStyle = bodyStyle;
//Auto-fit the columns
worksheet.UsedRange.AutofitColumns();
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Global
Styles.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("GlobalStyles.xlsx"
, "application/msexcel", stream);
}
}
```



Excel document with Global Styles

Apply Number Formats

Number Formats are codes that help to control the appearance of cell values especially numbers in an Excel document. Excel recognizes the numbers in various formats like:

- Number
- Currency
- Percentage
- DateTime
- Accounting
- Scientific
- Fraction and
- Text

This number format can be of maximum 4 parts, separated by semicolons. They are:

- Positive Numbers
- Negative Numbers
- Zeros
- Text

Each part is an individual number format. Default format is “General”, it means anything that will fit.

The following table shows various custom formatting codes:

Number Code	Description
General	General number format.
0 (zero)	Digit placeholder. This code pads the value with zeros to fill the format.

#	Digit placeholder. This code does not display extra zeros.
?	Digit placeholder. This code leaves a space for insignificant zeros but does not display them.
. (period)	Decimal placeholder. The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator.
%	Percentage placeholder. Multiplies by 100 and adds the % character.
, (comma)	Thousands separator. A comma followed by a placeholder (0 or #) scales the number by a thousand.
E+ E- e+ e-	Scientific notation.
Text Code	Description
\$ - + / () : space	These characters are displayed in the number. To display any other character, enclose the character in quotation marks or precede it with a backslash.
\character	<p>This code displays the succeeding character you specify.</p> <p>Note Typing !, ^, &, ', ~, {, }, =, automatically places a backslash in front of the character.</p>
"text"	This code displays the text.
*	<p>This code repeats the next character in the format to fill the column width.</p> <p>Note: Only one asterisk per section of a format is allowed.</p>
_ (underscore)	This code skips the width of the next character. This code is commonly used as "_" (without the quotation marks) to leave space for a closing parenthesis in a positive number format when the negative number format includes parentheses.

	This allows the values to line up at the decimal point.
@	Text placeholder.
Date Code	Description
m	Month as a number without leading zeros (1-12).
mm	Month as a number with leading zeros (01-12).
mmm	Month as an abbreviation (Jan - Dec).
mmmm	Unabbreviated Month (January - December).
d	Day without leading zeros (1-31).
dd	Day with leading zeros (01-31).
ddd	Week day as an abbreviation (Sun - Sat).
dddd	Unabbreviated week day (Sunday - Saturday).
yy	Year as a two-digit number (for example, 96).
yyyy	Year as a four-digit number (for example, 1996).
Time Code	Description
h	Hours as a number without leading zeros (0-23).
hh	Hours as a number with leading zeros (00-23).
m	Minutes as a number without leading zeros (0-59).
mm	Minutes as a number with leading zeros (00-59).
s	Seconds as a number without leading zeros (0-59).
ss	Seconds as a number with leading zeros (00-59).
AM/PM am/pm	Time based on the twelve-hour clock.
Miscellaneous Code	Description

<p>[BLACK], [BLUE], [CYAN], [GREEN], [MAGENTA], [RED], [WHITE], [YELLOW], [COLOR n]</p>	<p>These codes display the characters in the specified colors.</p> <p>Note: n is a value from 1 to 56 and refers to the nth color in the color palette.</p>
<p>[Condition value]</p>	<p>Condition may be , =, >=, <=, <> and value may be any number.</p> <p>Note: A number format may contain up to two conditions.</p>

XlsIO provides support for reading and writing various built-in and custom number formats in a cell by using the **NumberFormat** property of **IRange** interface.

The following code snippet illustrates how to set different number formats in a worksheet range.

C#

[illegible]

```

worksheet.Range["A9"].Text = "1.20";
worksheet.Range["B9"].Text = "0.00E+00";
worksheet.Range["C9"].NumberFormat = "0.00E+00";
worksheet.Range["C9"].Number = 1.20;
//Applying percentage format
worksheet.Range["A10"].Text = "1.20";
worksheet.Range["B10"].Text = "0.00%";
worksheet.Range["C10"].NumberFormat = "0.00%";
worksheet.Range["C10"].Number = 1.20;
//Applying date format
worksheet.Range["A11"].Text = new DateTime(2005, 12, 25).ToString();
worksheet.Range["B11"].Text = "m/d/yyyy";
worksheet.Range["C11"].NumberFormat = "m/d/yyyy";
worksheet.Range["C11"].DateTime = new DateTime(2005, 12, 25);
//Applying currency format
worksheet.Range["A12"].Text = "1.20";
worksheet.Range["B12"].Text = "$#,##0.00";
worksheet.Range["C12"].NumberFormat = "$#,##0.00";
worksheet.Range["C12"].Number = 1.20;
//Applying accounting format
worksheet.Range["A12"].Text = "234";
worksheet.Range["B12"].Text = "_($* #,##0_)";
worksheet.Range["C12"].NumberFormat = "_($* #,##0_)";
worksheet.Range["C12"].Number = 234;
//Fit column width to data
worksheet.UsedRange.AutoFitColumns();
workbook.SaveAs("NumberFormats.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
worksheet.Range("A1").Text = "DATA"
worksheet.Range("B1").Text = "NUMBER FORMAT APPLIED"
worksheet.Range("C1").Text = "RESULT"
Dim headingStyle As IStyle = workbook.Styles.Add("HeadingStyle")
headingStyle.Font.Bold = True
headingStyle.HorizontalAlignment = ExcelHAlign.HAlignCenter
worksheet.Range("A1:C1").CellStyle = headingStyle
'Applying different number formats
worksheet.Range("A2").Text = "1000000.00075"
worksheet.Range("B2").Text = "0.00"
worksheet.Range("C2").NumberFormat = "0.00"
worksheet.Range("C2").Number = 1000000.00075
worksheet.Range("A3").Text = "1000000.500"
worksheet.Range("B3").Text = "###,##"
worksheet.Range("C3").NumberFormat = "###,##"
worksheet.Range("C3").Number = 1000000.5
worksheet.Range("A5").Text = "10000"
worksheet.Range("B5").Text = "0.00"
worksheet.Range("C5").NumberFormat = "0.00"
worksheet.Range("C5").Number = 10000

```


[illegible]

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A1"].Text = "DATA";
    worksheet.Range["B1"].Text = "NUMBER FORMAT APPLIED";
    worksheet.Range["C1"].Text = "RESULT";
    IStyle headingStyle = workbook.Styles.Add("HeadingStyle");
    headingStyle.Font.Bold = true;
    headingStyle.HorizontalAlignment = ExcelHAlign.HAlignCenter;
    worksheet.Range["A1:C1"].CellStyle = headingStyle;
    //Applying different number formats
    worksheet.Range["A2"].Text = "1000000.00075";
    worksheet.Range["B2"].Text = "0.00";
    worksheet.Range["C2"].NumberFormat = "0.00";
    worksheet.Range["C2"].Number = 1000000.00075;
    worksheet.Range["A3"].Text = "1000000.500";
    worksheet.Range["B3"].Text = "###,##";
    worksheet.Range["C3"].NumberFormat = "###,##";
    worksheet.Range["C3"].Number = 1000000.500;
    worksheet.Range["A5"].Text = "10000";
    worksheet.Range["B5"].Text = "0.00";
    worksheet.Range["C5"].NumberFormat = "0.00";
    worksheet.Range["C5"].Number = 10000;
    worksheet.Range["A6"].Text = "-500";
    worksheet.Range["B6"].Text = "[Blue]#,##0";
    worksheet.Range["C6"].NumberFormat = "[Blue]#,##0";
    worksheet.Range["C6"].Number = -500;
    worksheet.Range["A7"].Text = "0.000000000000000000001234567890";
    worksheet.Range["B7"].Text = "0.00000000000000000000000000000000";
    worksheet.Range["C7"].NumberFormat = "0.00000000000000000000000000000000";
    worksheet.Range["C7"].Number = 0.000000000000000000001234567890;
    worksheet.Range["A9"].Text = "1.20";
    worksheet.Range["B9"].Text = "0.00E+00";
    worksheet.Range["C9"].NumberFormat = "0.00E+00";
    worksheet.Range["C9"].Number = 1.20;
    //Applying percentage format
    worksheet.Range["A10"].Text = "1.20";
    worksheet.Range["B10"].Text = "0.00%";
    worksheet.Range["C10"].NumberFormat = "0.00%";
    worksheet.Range["C10"].Number = 1.20;
    //Applying date format
    worksheet.Range["A11"].Text = new DateTime(2005, 12, 25).ToString();
    worksheet.Range["B11"].Text = "m/d/yyyy";
    worksheet.Range["C11"].NumberFormat = "m/d/yyyy";
    worksheet.Range["C11"].DateTime = new DateTime(2005, 12, 25);
    //Applying currency format
    worksheet.Range["A12"].Text = "1.20";
    worksheet.Range["B12"].Text = "$#,##0.00";
    worksheet.Range["C12"].NumberFormat = "$#,##0.00";
    worksheet.Range["C12"].Number = 1.20;
    //Applying accounting format
    worksheet.Range["A12"].Text = "234";
    worksheet.Range["B12"].Text = "_($* #,##0_)";
    worksheet.Range["C12"].NumberFormat = " ($* #,##0 )";
```

```
worksheet.Range["C12"].Number = 234;
//Fit column width to data
worksheet.UsedRange.AutoFitColumns();
//Saving the workbook as stream
FileStream stream = new FileStream("NumberFormats.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}
```

XAMARIN

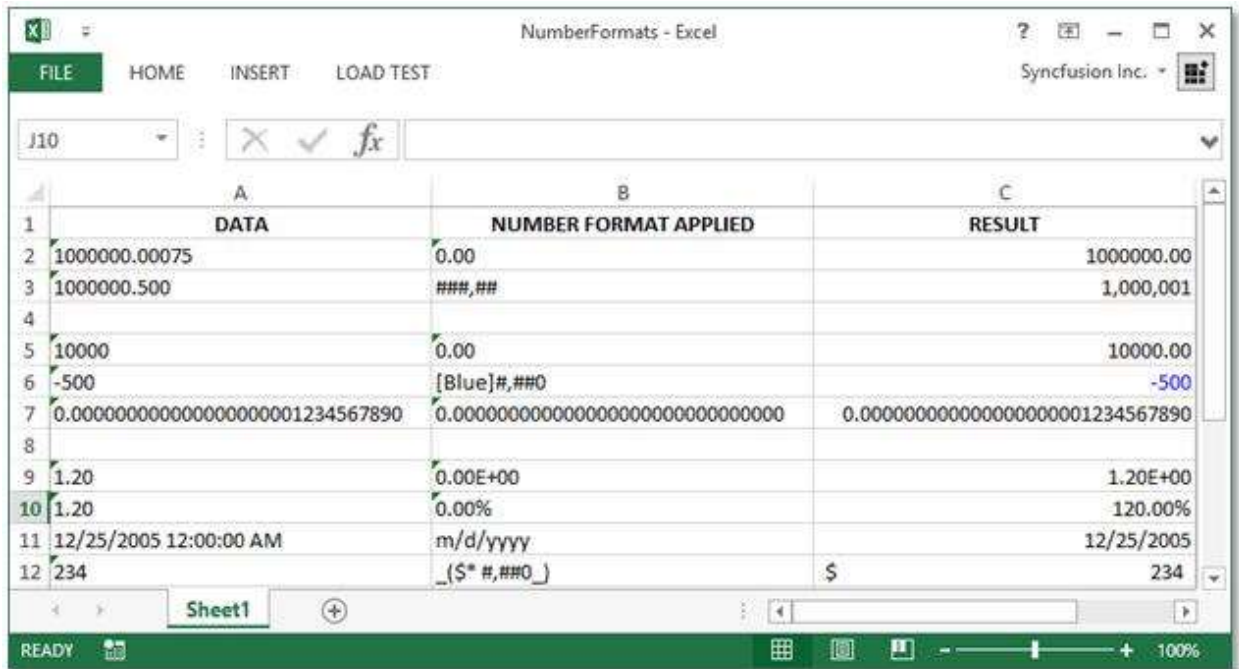
[illegible]

```

worksheet.Range["A11"].Text = new DateTime(2005, 12, 25).ToString();
worksheet.Range["B11"].Text = "m/d/yyyy";
worksheet.Range["C11"].NumberFormat = "m/d/yyyy";
worksheet.Range["C11"].DateTime = new DateTime(2005, 12, 25);
//Applying currency format
worksheet.Range["A12"].Text = "1.20";
worksheet.Range["B12"].Text = "$#,##0.00";
worksheet.Range["C12"].NumberFormat = "$#,##0.00";
worksheet.Range["C12"].Number = 1.20;
//Applying accounting format
worksheet.Range["A12"].Text = "234";
worksheet.Range["B12"].Text = "_(($* #,##0_)" ;
worksheet.Range["C12"].NumberFormat = "_(($* #,##0_)" ;
worksheet.Range["C12"].Number = 234;
//Fit column width to data
worksheet.UsedRange.AutofitColumns();
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Refer to the xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Number
Formats.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("NumberFormats.xlsx
", "application/msexcel", stream);
}
}

```

The screenshot of the previous code is shown as follows:



Access number format applied results at runtime

Cell values can be accessed as **Text**, **Number**, **DateTime** and **Formula** of **IRange** interface. In addition to this, there is an another property **DisplayText** in **IRange**, which returns a resultant value of a cell with its number format applied.

The following code example illustrates how to display the text of a cell.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    Application application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];

    //Set value to the cell
    worksheet.Range["C4"].Number = 1.20;

    //Set value to a cell
    worksheet.Range["B4"].Text = "$#,##0.00";

    //Get display text of the cell
    string text = worksheet.Range["B4"].DisplayText;

    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()  
Dim application As IApplication = excelEngine.Excel  
application.DefaultVersion = ExcelVersion.Excel2013  
Dim workbook As IWorkbook = application.Workbooks.Create(1)  
Dim worksheet As IWorksheet = workbook.Worksheets(0)  
'Set value to the cell  
worksheet.Range("C4").Number = 1.2
```

```

'Set value to a cell
worksheet.Range("B4").Text = "$#,##0.00"
'Get display text of the cell
Dim text As String = worksheet.Range("B4").DisplayText
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Set value to the cell
    worksheet.Range["C4"].Number = 1.20;
    //Set value to a cell
    worksheet.Range["B4"].Text = "$#,##0.00";
    //Get display text of the cell
    string text = worksheet.Range["B4"].DisplayText;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Set value to the cell
    worksheet.Range["C4"].Number = 1.20;
    //Set value to a cell
    worksheet.Range["B4"].Text = "$#,##0.00";
    //Get display text of the cell
    string text = worksheet.Range["B4"].DisplayText;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Set value to the cell
    worksheet.Range["C4"].Number = 1.20;
    //Set value to a cell
    worksheet.Range["B4"].Text = "$#,##0.00";
    //Get display text of the cell
    string text = worksheet.Range["B4"].DisplayText;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Refer to the xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
        "application/msexcel", stream);
    }
}

```

You can set **IWorkbook.DetectDateTimeInValue** property as 'false' with Value2 property, if you are sure that the given value is not of DateTime data type which improves time performance.

C#

```
workbook.DetectDateTimeInValue = false;
```

VB.NET

```
workbook.DetectDateTimeInValue = False
```

UWP

```
workbook.DetectDateTimeInValue = false;
```

ASP.NET CORE

```
workbook.DetectDateTimeInValue = false;
```

XAMARIN


```
workbook.DetectDateTimeInValue = false;
```

Hide Cell Content by setting Number Format

Essential XlsIO supports [hiding rows or columns](#) in a worksheet along with [hiding specific range](#). You can also hide a particular cell content by setting a specific number format to that cell.

Refer to the following complete code snippets.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Assign values to a range of cells in the worksheet
    worksheet.Range["A1:A10"].Text = "Hide Cell Content";
    //Apply number format for the cell to hide its content
    worksheet.Range["A5"].NumberFormat = ";;;";
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Assign values to a range of cells in the worksheet
worksheet.Range("A1:A10").Text = "Hide Cell Content"
'Apply number format for the cell to hide its content
worksheet.Range("A5").NumberFormat = ";;;";
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Assign values to a range of cells in the worksheet
    worksheet.Range["A1:A10"].Text = "Hide Cell Content";
    //Apply number format for the cell to hide its content
    worksheet.Range["A5"].NumberFormat = ";;;";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
```

```
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Assign values to a range of cells in the worksheet
    worksheet.Range["A1:A10"].Text = "Hide Cell Content";
    //Apply number format for the cell to hide its content
    worksheet.Range["A5"].NumberFormat = ";;;";
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Assign values to a range of cells in the worksheet
    worksheet.Range["A1:A10"].Text = "Hide Cell Content";
    //Apply number format for the cell to hide its content
    worksheet.Range["A5"].NumberFormat = ";;;";
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
        "application/msexcel", stream);
    }
}
```

```
}
```

Apply Cell Text Alignment

The XlsIO supports the following alignment options:

- Horizontal Alignment
- Vertical Alignment
- Indentation
- Orientation
- Text Direction

Horizontal Alignment

This code snippet aligns the cell content horizontally.

C#

```
//Text Alignment Setting (Horizontal Alignment)  
worksheet.Range["A2"].CellStyle.HorizontalAlign =  
ExcelHAlign.HAlignCenter;
```

VB.NET

```
'Text Alignment Setting (Horizontal Alignment)  
worksheet.Range("A2").CellStyle.HorizontalAlign =  
ExcelHAlign.HAlignCenter
```

UWP

```
//Text Alignment Setting (Horizontal Alignment)  
worksheet.Range["A2"].CellStyle.HorizontalAlign =  
ExcelHAlign.HAlignCenter;
```

ASP.NET CORE

```
//Text Alignment Setting (Horizontal Alignment)  
worksheet.Range["A2"].CellStyle.HorizontalAlign =  
ExcelHAlign.HAlignCenter;
```

XAMARIN

```
//Text Alignment Setting (Horizontal Alignment)  
worksheet.Range["A2"].CellStyle.HorizontalAlign =  
ExcelHAlign.HAlignCenter;
```

Vertical Alignment

This code snippet aligns the cell content vertically.

C#

```
//Text Alignment Setting (Vertical Alignment)
```

```
worksheet.Range["B2"].CellStyle.VerticalAlignment =  
ExcelVAlign.VAlignBottom;
```

VB.NET

```
'Text Alignment Setting (Vertical Alignment)  
worksheet.Range("B2").CellStyle.VerticalAlignment = ExcelVAlign.VAlignBottom
```

UWP

```
//Text Alignment Setting (Vertical Alignment)  
worksheet.Range["B2"].CellStyle.VerticalAlignment =  
ExcelVAlign.VAlignBottom;
```

ASP.NET CORE

```
//Text Alignment Setting (Vertical Alignment)  
worksheet.Range["B2"].CellStyle.VerticalAlignment =  
ExcelVAlign.VAlignBottom;
```

XAMARIN

```
//Text Alignment Setting (Vertical Alignment)  
worksheet.Range["B2"].CellStyle.VerticalAlignment =  
ExcelVAlign.VAlignBottom;
```

Indentation

This allows to set the cell content either to move it closer to the cell border or to move it farther away from cell border.

C#

```
//Text Indent Setting  
worksheet.Range["C6"].CellStyle.IndentLevel = 6;
```

VB.NET

```
'Text Indent Setting  
worksheet.Range("C6").CellStyle.IndentLevel = 6
```

UWP

```
//Text Indent Setting  
worksheet.Range["C6"].CellStyle.IndentLevel = 6;
```

ASP.NET CORE

```
//Text Indent Setting  
worksheet.Range["C6"].CellStyle.IndentLevel = 6;
```

XAMARIN

```
//Text Indent Setting  
worksheet.Range["C6"].CellStyle.IndentLevel = 6;
```

Orientation

This helps to rotate the cell text diagonally or vertically. The text orientation can be set by using the **Rotation** property as shown as follows.

C#

```
//Text Orientation Settings  
worksheet.Range["C2"].CellStyle.Rotation = 60;
```

VB.NET

```
'Text Orientation Settings  
worksheet.Range("C2").CellStyle.Rotation = 60
```

UWP

```
//Text Orientation Settings  
worksheet.Range["C2"].CellStyle.Rotation = 60;
```

ASP.NET CORE

```
//Text Orientation Settings  
worksheet.Range["C2"].CellStyle.Rotation = 60;
```

XAMARIN

```
//Text Orientation Settings  
worksheet.Range["C2"].CellStyle.Rotation = 60;
```

Text Direction

You can specify the text direction by using the **ReadingOrder** property as shown as follows.

C#

```
//Text Direction Setting  
worksheet.Range["D2"].CellStyle.ReadingOrder =  
ExcelReadingOrderType.LeftToRight;
```

VB.NET

```
'Text Direction Setting  
worksheet.Range("D2").CellStyle.ReadingOrder =  
ExcelReadingOrderType.LeftToRight
```

UWP

```
//Text Direction Setting
```

```
worksheet.Range["D2"].CellStyle.ReadingOrder =
ExcelReadingOrderType.LeftToRight;
```

ASP.NET CORE

```
//Text Direction Setting
worksheet.Range["D2"].CellStyle.ReadingOrder =
ExcelReadingOrderType.LeftToRight;
```

XAMARIN

```
//Text Direction Setting
worksheet.Range["D2"].CellStyle.ReadingOrder =
ExcelReadingOrderType.LeftToRight;
```

The following is the complete code snippet illustrating the previous options.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A2"].Text = "HAlignCenter";
    worksheet.Range["A4"].Text = "HAlignFill";
    worksheet.Range["A6"].Text = "HAlignRight";
    worksheet.Range["A8"].Text = "HAlignCenterAcrossSelection";
    worksheet.Range["B2"].Text = "VAlignCenter";
    worksheet.Range["B4"].Text = "VAlignFill";
    worksheet.Range["B6"].Text = "VAlignTop";
    worksheet.Range["B8"].Text = "VAlignCenterAcrossSelection";
    worksheet.Range["C2"].Text = "Text Rotation to 60 degree";
    worksheet.Range["C4"].Text = "Text Rotation to 90 degree";
    worksheet.Range["C6"].Text = "Indent level is 6";
    worksheet.Range["D2"].Text = "Text Direction(LeftToRight)";
    worksheet.Range["D3"].Text = "Text Direction(RightToLeft)";
    worksheet.Range["D4"].Text = "Text Direction(Context)";
    //Text Alignment Setting (Horizontal Alignment)
    worksheet.Range["A2"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignCenter;
    worksheet.Range["A4"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignFill;
    worksheet.Range["A6"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignRight;
    worksheet.Range["A8"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignCenterAcrossSelection;
    //Text Alignment Setting (Vertical Alignment)
    worksheet.Range["B2"].CellStyle.VerticalAlignment =
ExcelVAlign.VAlignBottom;
    worksheet.Range["B4"].CellStyle.VerticalAlignment =
ExcelVAlign.VAlignCenter;
    worksheet.Range["B6"].CellStyle.VerticalAlignment = ExcelVAlign.VAlignTop;
```

```

worksheet.Range["B8"].CellStyle.VerticalAlignment =
ExcelVAlign.VAlignDistributed;
//Text Orientation Settings
worksheet.Range["C2"].CellStyle.Rotation = 60;
worksheet.Range["C4"].CellStyle.Rotation = 90;
//Text Indent Setting
worksheet.Range["C6"].CellStyle.IndentLevel = 6;
//Text Direction Setting
worksheet.Range["D2"].CellStyle.ReadingOrder =
ExcelReadingOrderType.LeftToRight;
worksheet.Range["D3"].CellStyle.ReadingOrder =
ExcelReadingOrderType.RightToLeft;
worksheet.Range["D4"].CellStyle.ReadingOrder =
ExcelReadingOrderType.Context;
worksheet.UsedRange.AutofitColumns();
worksheet.UsedRange.AutofitRows();
workbook.SaveAs("Book1.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
worksheet.Range("A2").Text = "HAlignCenter"
worksheet.Range("A4").Text = "HAlignFill"
worksheet.Range("A6").Text = "HAlignRight"
worksheet.Range("A8").Text = "HAlignCenterAcrossSelection"
worksheet.Range("B2").Text = "VAlignCenter"
worksheet.Range("B4").Text = "VAlignFill"
worksheet.Range("B6").Text = "VAlignTop"
worksheet.Range("B8").Text = "VAlignCenterAcrossSelection"
worksheet.Range("C2").Text = "Text Rotation to 60 degree"
worksheet.Range("C4").Text = "Text Rotation to 90 degree"
worksheet.Range("C6").Text = "Indent level is 6"
worksheet.Range("D2").Text = "Text Direction(LeftToRight)"
worksheet.Range("D3").Text = "Text Direction(RightToLeft)"
worksheet.Range("D4").Text = "Text Direction(Context)"
'Text Alignment Setting (Horizontal Alignment)
worksheet.Range("A2").CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignCenter
worksheet.Range("A4").CellStyle.HorizontalAlignment = ExcelHAlign.HAlignFill
worksheet.Range("A6").CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignRight
worksheet.Range("A8").CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignCenterAcrossSelection
'Text Alignment Setting (Vertical Alignment)
worksheet.Range("B2").CellStyle.VerticalAlignment = ExcelVAlign.VAlignBottom
worksheet.Range("B4").CellStyle.VerticalAlignment = ExcelVAlign.VAlignCenter
worksheet.Range("B6").CellStyle.VerticalAlignment = ExcelVAlign.VAlignTop
worksheet.Range("B8").CellStyle.VerticalAlignment =
ExcelVAlign.VAlignDistributed
'Text Orientation Settings
worksheet.Range("C2").CellStyle.Rotation = 60

```

```

worksheet.Range("C4").CellStyle.Rotation = 90
'Text Indent Setting
worksheet.Range("C6").CellStyle.IndentLevel = 6
'Text Direction Setting
worksheet.Range("D2").CellStyle.ReadingOrder =
ExcelReadingOrderType.LeftToRight
worksheet.Range("D3").CellStyle.ReadingOrder =
ExcelReadingOrderType.RightToLeft
worksheet.Range("D4").CellStyle.ReadingOrder = ExcelReadingOrderType.Context
worksheet.UsedRange.AutofitColumns()
worksheet.UsedRange.AutofitRows()
workbook.SaveAs("Book1.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A2"].Text = "HAlignCenter";
    worksheet.Range["A4"].Text = "HAlignFill";
    worksheet.Range["A6"].Text = "HAlignRight";
    worksheet.Range["A8"].Text = "HAlignCenterAcrossSelection";
    worksheet.Range["B2"].Text = "VAlignCenter";
    worksheet.Range["B4"].Text = "VAlignFill";
    worksheet.Range["B6"].Text = "VAlignTop";
    worksheet.Range["B8"].Text = "VAlignCenterAcrossSelection";
    worksheet.Range["C2"].Text = "Text Rotation to 60 degree";
    worksheet.Range["C4"].Text = "Text Rotation to 90 degree";
    worksheet.Range["C6"].Text = "Indent level is 6";
    worksheet.Range["D2"].Text = "Text Direction(LeftToRight)";
    worksheet.Range["D3"].Text = "Text Direction(RightToLeft)";
    worksheet.Range["D4"].Text = "Text Direction(Context)";
    //Text Alignment Setting (Horizontal Alignment)
    worksheet.Range["A2"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignCenter;
    worksheet.Range["A4"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignFill;
    worksheet.Range["A6"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignRight;
    worksheet.Range["A8"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignCenterAcrossSelection;
    //Text Alignment Setting (Vertical Alignment)
    worksheet.Range["B2"].CellStyle.VerticalAlignment =
ExcelVAlign.VAlignBottom;
    worksheet.Range["B4"].CellStyle.VerticalAlignment =
ExcelVAlign.VAlignCenter;
    worksheet.Range["B6"].CellStyle.VerticalAlignment = ExcelVAlign.VAlignTop;
    worksheet.Range["B8"].CellStyle.VerticalAlignment =
ExcelVAlign.VAlignDistributed;
    //Text Orientation Settings
    worksheet.Range["C2"].CellStyle.Rotation = 60;
    worksheet.Range["C4"].CellStyle.Rotation = 90;
}

```



```

//Text Indent Setting
worksheet.Range["C6"].CellStyle.IndentLevel = 6;
//Text Direction Setting
worksheet.Range["D2"].CellStyle.ReadingOrder =
ExcelReadingOrderType.LeftToRight;
worksheet.Range["D3"].CellStyle.ReadingOrder =
ExcelReadingOrderType.RightToLeft;
worksheet.Range["D4"].CellStyle.ReadingOrder =
ExcelReadingOrderType.Context;
worksheet.UsedRange.AutofitColumns();
worksheet.UsedRange.AutofitRows();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Book1";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A2"].Text = "HAlignCenter";
    worksheet.Range["A4"].Text = "HAlignFill";
    worksheet.Range["A6"].Text = "HAlignRight";
    worksheet.Range["A8"].Text = "HAlignCenterAcrossSelection";
    worksheet.Range["B2"].Text = "VAlignCenter";
    worksheet.Range["B4"].Text = "VAlignFill";
    worksheet.Range["B6"].Text = "VAlignTop";
    worksheet.Range["B8"].Text = "VAlignCenterAcrossSelection";
    worksheet.Range["C2"].Text = "Text Rotation to 60 degree";
    worksheet.Range["C4"].Text = "Text Rotation to 90 degree";
    worksheet.Range["C6"].Text = "Indent level is 6";
    worksheet.Range["D2"].Text = "Text Direction(LeftToRight)";
    worksheet.Range["D3"].Text = "Text Direction(RightToLeft)";
    worksheet.Range["D4"].Text = "Text Direction(Context)";
    //Text Alignment Setting (Horizontal Alignment)
    worksheet.Range["A2"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignCenter;
    worksheet.Range["A4"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignFill;
    worksheet.Range["A6"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignRight;
    worksheet.Range["A8"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignCenterAcrossSelection;
    //Text Alignment Setting (Vertical Alignment)
}

```

```

worksheet.Range["B2"].CellStyle.VerticalAlignment =
ExcelVAlign.VAlignBottom;
worksheet.Range["B4"].CellStyle.VerticalAlignment =
ExcelVAlign.VAlignCenter;
worksheet.Range["B6"].CellStyle.VerticalAlignment = ExcelVAlign.VAlignTop;
worksheet.Range["B8"].CellStyle.VerticalAlignment =
ExcelVAlign.VAlignDistributed;
//Text Orientation Settings
worksheet.Range["C2"].CellStyle.Rotation = 60;
worksheet.Range["C4"].CellStyle.Rotation = 90;
//Text Indent Setting
worksheet.Range["C6"].CellStyle.IndentLevel = 6;
//Text Direction Setting
worksheet.Range["D2"].CellStyle.ReadingOrder =
ExcelReadingOrderType.LeftToRight;
worksheet.Range["D3"].CellStyle.ReadingOrder =
ExcelReadingOrderType.RightToLeft;
worksheet.Range["D4"].CellStyle.ReadingOrder =
ExcelReadingOrderType.Context;
worksheet.UsedRange.AutofitColumns();
worksheet.UsedRange.AutofitRows();
//Saving the workbook as stream
FileStream stream = new FileStream("Book1.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A2"].Text = "HAlignCenter";
    worksheet.Range["A4"].Text = "HAlignFill";
    worksheet.Range["A6"].Text = "HAlignRight";
    worksheet.Range["A8"].Text = "HAlignCenterAcrossSelection";
    worksheet.Range["B2"].Text = "VAlignCenter";
    worksheet.Range["B4"].Text = "VAlignFill";
    worksheet.Range["B6"].Text = "VAlignTop";
    worksheet.Range["B8"].Text = "VAlignCenterAcrossSelection";
    worksheet.Range["C2"].Text = "Text Rotation to 60 degree";
    worksheet.Range["C4"].Text = "Text Rotation to 90 degree";
    worksheet.Range["C6"].Text = "Indent level is 6";
    worksheet.Range["D2"].Text = "Text Direction(LeftToRight)";
    worksheet.Range["D3"].Text = "Text Direction(RightToLeft)";
    worksheet.Range["D4"].Text = "Text Direction(Context)";
    //Text Alignment Setting (Horizontal Alignment)
    worksheet.Range["A2"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignCenter;
    worksheet.Range["A4"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignFill;
}

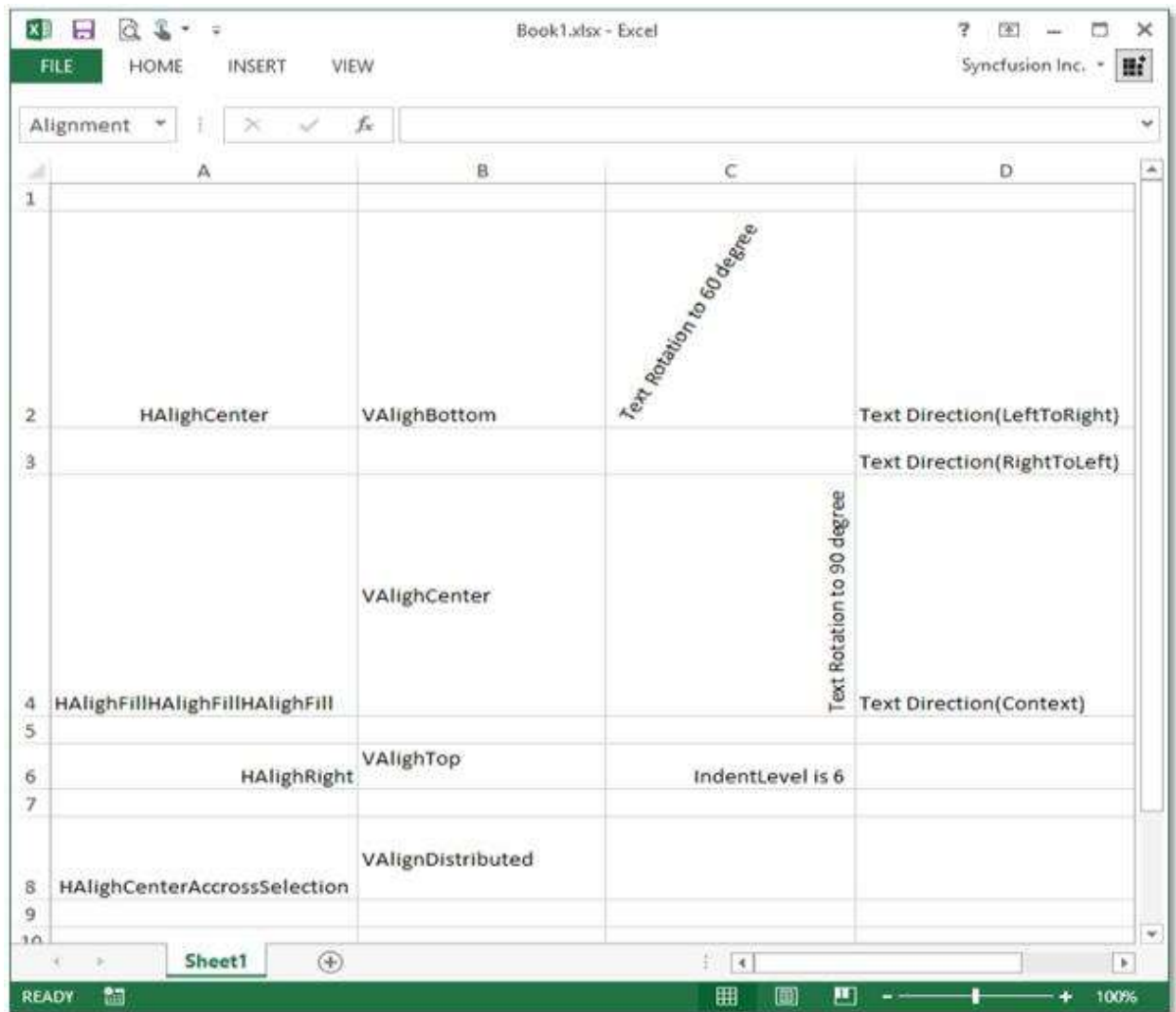
```

```

worksheet.Range["A6"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignRight;
worksheet.Range["A8"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignCenterAcrossSelection;
//Text Alignment Setting (Vertical Alignment)
worksheet.Range["B2"].CellStyle.VerticalAlignment =
ExcelVAlign.VAlignBottom;
worksheet.Range["B4"].CellStyle.VerticalAlignment =
ExcelVAlign.VAlignCenter;
worksheet.Range["B6"].CellStyle.VerticalAlignment = ExcelVAlign.VAlignTop;
worksheet.Range["B8"].CellStyle.VerticalAlignment =
ExcelVAlign.VAlignDistributed;
//Text Orientation Settings
worksheet.Range["C2"].CellStyle.Rotation = 60;
worksheet.Range["C4"].CellStyle.Rotation = 90;
//Text Indent Setting
worksheet.Range["C6"].CellStyle.IndentLevel = 6;
//Text Direction Setting
worksheet.Range["D2"].CellStyle.ReadingOrder =
ExcelReadingOrderType.LeftToRight;
worksheet.Range["D3"].CellStyle.ReadingOrder =
ExcelReadingOrderType.RightToLeft;
worksheet.Range["D4"].CellStyle.ReadingOrder =
ExcelReadingOrderType.Context;
worksheet.UsedRange.AutofitColumns();
worksheet.UsedRange.AutofitRows();
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Book1.
xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Book1.xlsx",
"application/msexcel", stream);
}
}

```

The following screenshot is the output of previous code:



Merging and Un-Merging Cells

The cells can be merged using the **Merge()** method in **IRange** as shown as follows.

C#

```
//Merging Cells from A16 to C16
worksheet.Range["A16:C16"].Merge();
```

VB.NET

```
'Merging Cells from A16 to C16
worksheet.Range("A16:C16").Merge()
```

UWP

```
//Merging Cells from A16 to C16
worksheet.Range["A16:C16"].Merge();
```

ASP.NET CORE

```
//Merging Cells from A16 to C16  
worksheet.Range["A16:C16"].Merge();
```

XAMARIN

```
//Merging Cells from A16 to C16  
worksheet.Range["A16:C16"].Merge();
```

Merged cells can be unmerged using the **UnMerge()** method in **IRange** as shown below.

C#

```
//Un-Merging merged cells from A16 to C16  
worksheet.Range["A16:C16"].UnMerge();
```

VB.NET

```
'Un-Merging merged cells from A16 to C16  
worksheet.Range["A16:C16"].UnMerge();
```

UWP

```
//Un-Merging merged cells from A16 to C16  
worksheet.Range["A16:C16"].UnMerge();
```

ASP.NET CORE

```
//Un-Merging merged cells from A16 to C16  
worksheet.Range["A16:C16"].UnMerge();
```

XAMARIN

```
//Un-Merging merged cells from A16 to C16  
worksheet.Range["A16:C16"].UnMerge();
```

The below code shows merging and unmerging worksheet cells.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())  
{  
    IApplication application = excelEngine.Excel;  
    application.DefaultVersion = ExcelVersion.Excel2013;  
    IWorkbook workbook = application.Workbooks.Create(1);  
    IWorksheet worksheet = workbook.Worksheets[0];  
    //Merging cells  
    worksheet.Range["A16:C16"].Merge();  
    //Un-Merging merged cells  
    worksheet.Range["A16:C16"].UnMerge();  
    workbook.SaveAs("MergingUnMerging.xlsx");  
}
```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create()
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Merging cells
worksheet.Range("A16:C16").Merge()
'Un-Merging merged cells
worksheet.Range("A16:C16").UnMerge()
workbook.SaveAs("MergingUnMerging.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Merging cells
    worksheet.Range["A16:C16"].Merge();
    //Un-Merging merged cells
    worksheet.Range["A16:C16"].UnMerge();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "MergingUnMerging";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Merging cells
    worksheet.Range["A16:C16"].Merge();
    //Un-Merging merged cells
    worksheet.Range["A16:C16"].UnMerge();
    //Saving the workbook as stream
    FileStream stream = new FileStream("MergingUnMerging.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Merging cells
    worksheet.Range["A16:C16"].Merge();
    //Un-Merging merged cells
    worksheet.Range["A16:C16"].UnMerge();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("MergingUnMerging.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("MergingUnMerging.xlsx", "application/msexcel", stream);
    }
}

```

Apply Wrap Text

If a cell content is too wide to fit a column and do not want to split over into adjacent cells, you can use the **WrapText** property. This will set the content within the cell border. The following code snippet illustrates this behavior.

Note: Applying wrap-text will not auto-fit the rows by default. It is recommended to [auto-fit](#) manually.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A2"].Text = "First Sentence is wrapped";
    worksheet.Range["B2"].Text = "Second Sentence is wrapped";
    worksheet.Range["C2"].Text = "Third Sentence is wrapped";
    //Applying Wrap-text
    worksheet.Range["A2:C2"].WrapText = true;
    workbook.SaveAs("WrapText.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
worksheet.Range("A2").Text = "First Sentence is wrapped"
worksheet.Range("B2").Text = "Second Sentence is wrapped"
worksheet.Range("C2").Text = "Third Sentence is wrapped"
'Applying wrap-text
worksheet.Range("A2:C2").WrapText = True
workbook.SaveAs("WrapText.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A2"].Text = "First Sentence is wrapped";
    worksheet.Range["B2"].Text = "Second Sentence is wrapped";
    worksheet.Range["C2"].Text = "Third Sentence is wrapped";
    //Applying Wrap-text
    worksheet.Range["A2:C2"].WrapText = true;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "WrapText";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A2"].Text = "First Sentence is wrapped";
    worksheet.Range["B2"].Text = "Second Sentence is wrapped";
    worksheet.Range["C2"].Text = "Third Sentence is wrapped";
    //Applying Wrap-text
    worksheet.Range["A2:C2"].WrapText = true;
    //Saving the workbook as stream
    FileStream stream = new FileStream("WrapText.xlsx", FileMode.Create,
    FileAccess.ReadWrite);

```



```
workbook.SaveAs(stream);
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A2"].Text = "First Sentence is wrapped";
    worksheet.Range["B2"].Text = "Second Sentence is wrapped";
    worksheet.Range["C2"].Text = "Third Sentence is wrapped";
    //Applying Wrap-text
    worksheet.Range["A2:C2"].WrapText = true;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("WrapText.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("WrapText.xlsx", "application/msexcel", stream);
    }
}
```

Auto-Fit Rows or Columns

Cell dimensions can be auto-sized to its content dynamically to make its content visible.

The following code shows how to auto-size row height and column width to its cell content.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Auto-fit rows
    worksheet.Range["A2"].Text = "Fit the content to row";
    worksheet.Range["A2"].WrapText = true;
    worksheet.Range["A2"].AutofitRows();
    //Auto-fit columns
```

```
worksheet.Range["B4"].Text = "Fit the content to column";
worksheet.Range["B4"].AutofitColumns();
workbook.SaveAs("AutoFit.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Auto-fit rows
worksheet.Range("A2").Text = "Fit the content to row"
worksheet.Range("A2").WrapText = True
worksheet.Range("A2").AutofitRows()
'Auto-fit columns
worksheet.Range("B4").Text = "Fit the content to column"
worksheet.Range("B4").AutofitColumns()
workbook.SaveAs("AutoFit.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Auto-fit rows
    worksheet.Range["A2"].Text = "Fit the content to row";
    worksheet.Range["A2"].WrapText = true;
    worksheet.Range["A2"].AutofitRows();
    //Auto-fit columns
    worksheet.Range["B4"].Text = "Fit the content to column";
    worksheet.Range["B4"].AutofitColumns();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "AutoFit";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
```

```

IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Auto-fit rows
worksheet.Range["A2"].Text = "Fit the content to row";
worksheet.Range["A2"].WrapText = true;
worksheet.Range["A2"].AutofitRows();
//Auto-fit columns
worksheet.Range["B4"].Text = "Fit the content to column";
worksheet.Range["B4"].AutofitColumns();
//Saving the workbook as stream
FileStream stream = new FileStream("AutoFit.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Auto-fit rows
    worksheet.Range["A2"].Text = "Fit the content to row";
    worksheet.Range["A2"].WrapText = true;
    worksheet.Range["A2"].AutofitRows();
    //Auto-fit columns
    worksheet.Range["B4"].Text = "Fit the content to column";
    worksheet.Range["B4"].AutofitColumns();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("AutoFi
t.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("AutoFit.xlsx",
"application/msexcel", stream);
    }
}

```

The output of the previous code is shown as follows.

Apply Font Settings

The appearance of a text can be controlled by font settings of a cell. These settings can be done by using the **Font** property in **CellStyle**. Refer to the following code.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding text for a range
    worksheet.Range["A1:B6"].Text = "Hello World";
    //Setting Font Type
    worksheet.Range["A1"].CellStyle.Font.FontName = "Arial Black";
    worksheet.Range["A3"].CellStyle.Font.FontName = "Castellar";
    //Setting Font Styles
    worksheet.Range["A2"].CellStyle.Font.Bold = true;
    worksheet.Range["A4"].CellStyle.Font.Italic = true;
    //Setting Font Size
    worksheet.Range["A5"].CellStyle.Font.Size = 18;
    //Setting Font Effects
    worksheet.Range["A6"].CellStyle.Font.Strikethrough = true;
    worksheet.Range["B3"].CellStyle.Font.Subscript = true;
    worksheet.Range["B5"].CellStyle.Font.Superscript = true;
    //Setting UnderLine Types
    worksheet.Range["B1"].CellStyle.Font.Underline = ExcelUnderline.Double;
    worksheet.Range["B2"].CellStyle.Font.Underline = ExcelUnderline.Single;
    worksheet.Range["B4"].CellStyle.Font.Underline =
    ExcelUnderline.DoubleAccounting;
    worksheet.Range["B6"].CellStyle.Font.Underline =
    ExcelUnderline.SingleAccounting;
    //Setting Font Color
    worksheet.Range["B6"].CellStyle.Font.Color = ExcelKnownColors.Green;
    worksheet.UsedRange.AutofitColumns();
    worksheet.UsedRange.AutofitRows();
    workbook.SaveAs("FontSettings.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding text for a range
worksheet.Range("A1:B6").Text = "Hello World"
'Setting Font Type
worksheet.Range("A1").CellStyle.Font.FontName = "Arial Black"
worksheet.Range("A3").CellStyle.Font.FontName = "Castellar"
'Setting Font Styles
worksheet.Range("A2").CellStyle.Font.Bold = True
worksheet.Range("A4").CellStyle.Font.Italic = True
'Setting Font Size
```

```

worksheet.Range("A5").CellStyle.Font.Size = 18
'Setting Font Effects
worksheet.Range("A6").CellStyle.Font.Strikethrough = True
worksheet.Range("B3").CellStyle.Font.Subscript = True
worksheet.Range("B5").CellStyle.Font.Superscript = True
'Setting UnderLine Types
worksheet.Range("B1").CellStyle.Font.Underline = ExcelUnderline.Double
worksheet.Range("B2").CellStyle.Font.Underline = ExcelUnderline.Single
worksheet.Range("B4").CellStyle.Font.Underline =
ExcelUnderline.DoubleAccounting
worksheet.Range("B6").CellStyle.Font.Underline =
ExcelUnderline.SingleAccounting
'Setting Font Color
worksheet.Range("B6").CellStyle.Font.Color = ExcelKnownColors.Green
worksheet.UsedRange.AutofitColumns()
worksheet.UsedRange.AutofitRows()
workbook.SaveAs("FontSettings.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding text for a range
    worksheet.Range["A1:B6"].Text = "Hello World";
    //Setting Font Type
    worksheet.Range["A1"].CellStyle.Font.FontName = "Arial Black";
    worksheet.Range["A3"].CellStyle.Font.FontName = "Castellar";
    //Setting Font Styles
    worksheet.Range["A2"].CellStyle.Font.Bold = true;
    worksheet.Range["A4"].CellStyle.Font.Italic = true;
    //Setting Font Size
    worksheet.Range["A5"].CellStyle.Font.Size = 18;
    //Setting Font Effects
    worksheet.Range["A6"].CellStyle.Font.Strikethrough = true;
    worksheet.Range["B3"].CellStyle.Font.Subscript = true;
    worksheet.Range["B5"].CellStyle.Font.Superscript = true;
    //Setting UnderLine Types
    worksheet.Range["B1"].CellStyle.Font.Underline = ExcelUnderline.Double;
    worksheet.Range["B2"].CellStyle.Font.Underline = ExcelUnderline.Single;
    worksheet.Range["B4"].CellStyle.Font.Underline =
ExcelUnderline.DoubleAccounting;
    worksheet.Range["B6"].CellStyle.Font.Underline =
ExcelUnderline.SingleAccounting;
    //Setting Font Color
    worksheet.Range["B6"].CellStyle.Font.Color = ExcelKnownColors.Green;
    worksheet.UsedRange.AutofitColumns();
    worksheet.UsedRange.AutofitRows();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "FontSettings";
}

```

```

savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding text for a range
    worksheet.Range["A1:B6"].Text = "Hello World";
    //Setting Font Type
    worksheet.Range["A1"].CellStyle.Font.FontName = "Arial Black";
    worksheet.Range["A3"].CellStyle.Font.FontName = "Castellar";
    //Setting Font Styles
    worksheet.Range["A2"].CellStyle.Font.Bold = true;
    worksheet.Range["A4"].CellStyle.Font.Italic = true;
    //Setting Font Size
    worksheet.Range["A5"].CellStyle.Font.Size = 18;
    //Setting Font Effects
    worksheet.Range["A6"].CellStyle.Font.Strikethrough = true;
    worksheet.Range["B3"].CellStyle.Font.Subscript = true;
    worksheet.Range["B5"].CellStyle.Font.Superscript = true;
    //Setting UnderLine Types
    worksheet.Range["B1"].CellStyle.Font.Underline = ExcelUnderline.Double;
    worksheet.Range["B2"].CellStyle.Font.Underline = ExcelUnderline.Single;
    worksheet.Range["B4"].CellStyle.Font.Underline =
    ExcelUnderline.DoubleAccounting;
    worksheet.Range["B6"].CellStyle.Font.Underline =
    ExcelUnderline.SingleAccounting;
    //Setting Font Color
    worksheet.Range["B6"].CellStyle.Font.Color = ExcelKnownColors.Green;
    worksheet.UsedRange.AutofitColumns();
    worksheet.UsedRange.AutofitRows();
    //Saving the workbook as stream
    FileStream stream = new FileStream("FontSettings.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];

```

```
//Adding text for a range
worksheet.Range["A1:B6"].Text = "Hello World";
//Setting Font Type
worksheet.Range["A1"].CellStyle.Font.FontName = "Arial Black";
worksheet.Range["A3"].CellStyle.Font.FontName = "Castellar";
//Setting Font Styles
worksheet.Range["A2"].CellStyle.Font.Bold = true;
worksheet.Range["A4"].CellStyle.Font.Italic = true;
//Setting Font Size
worksheet.Range["A5"].CellStyle.Font.Size = 18;
//Setting Font Effects
worksheet.Range["A6"].CellStyle.Font.Strikethrough = true;
worksheet.Range["B3"].CellStyle.Font.Subscript = true;
worksheet.Range["B5"].CellStyle.Font.Superscript = true;
//Setting UnderLine Types
worksheet.Range["B1"].CellStyle.Font.Underline = ExcelUnderline.Double;
worksheet.Range["B2"].CellStyle.Font.Underline = ExcelUnderline.Single;
worksheet.Range["B4"].CellStyle.Font.Underline =
ExcelUnderline.DoubleAccounting;
worksheet.Range["B6"].CellStyle.Font.Underline =
ExcelUnderline.SingleAccounting;
//Setting Font Color
worksheet.Range["B6"].CellStyle.Font.Color = ExcelKnownColors.Green;
worksheet.UsedRange.AutofitColumns();
worksheet.UsedRange.AutofitRows();
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("FontSe
ttings.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("FontSettings.xlsx"
, "application/msexcel", stream);
}
}
```

The output of the previous code is shown as follows.

	A	B
1	Hello World	Hello World
2	Hello World	Hello World
3	HELLO WORLD	Hello World
4	Hello World	Hello World
5	Hello World	Hello World
6	Hello World	Hello World
7		

Apply Color Settings

Colors give enhancement to cell values to highlight the data. These color settings in a cell are differentiated as BackColor, ForeColor, and PatternColor.

Back Color settings

Back color of a cell can be set using the **ColorIndex** property of **CellStyle** as shown as follows.

C#

```
//Apply cell back color
worksheet.Range["A1"].CellStyle.ColorIndex = ExcelKnownColors.Aqua;
```

VB.NET

```
'Apply cell back color
worksheet.Range("A1").CellStyle.ColorIndex = ExcelKnownColors.Aqua
```

UWP

```
//Apply cell back color
worksheet.Range["A1"].CellStyle.ColorIndex = ExcelKnownColors.Aqua;
```

ASP.NET CORE

```
//Apply cell back color
worksheet.Range["A1"].CellStyle.ColorIndex = ExcelKnownColors.Aqua;
```

XAMARIN

```
//Apply cell back color
worksheet.Range["A1"].CellStyle.ColorIndex = ExcelKnownColors.Aqua;
```

Fore Color Settings

Fore color of a cell can be set using the **PatternColorIndex** property of **CellStyle** as shown as follows.

C#

```
//Apply cell fore color
worksheet.Range["A2"].CellStyle.PatternColorIndex = ExcelKnownColors.Green;
```


VB.NET

```
'Apply cell fore color  
worksheet.Range("A2").CellStyle.PatternColorIndex = ExcelKnownColors.Green
```

UWP

```
//Apply cell fore color  
worksheet.Range["A2"].CellStyle.PatternColorIndex = ExcelKnownColors.Green;
```

ASP.NET CORE

```
//Apply cell fore color  
worksheet.Range["A2"].CellStyle.PatternColorIndex = ExcelKnownColors.Green;
```

XAMARIN

```
//Apply cell fore color  
worksheet.Range["A2"].CellStyle.PatternColorIndex = ExcelKnownColors.Green;
```

Pattern Settings

Excel provides various pattern styles for highlighting the cells. These patterns can be applied using the **FillPattern** property of **CellStyle** as shown as follows.

C#

```
//Apply cell pattern  
worksheet.Range["A2"].CellStyle.FillPattern = ExcelPattern.Angle;
```

VB.NET

```
'Apply cell pattern  
worksheet.Range("A2").CellStyle.FillPattern = ExcelPattern.Angle
```

UWP

```
//Apply cell pattern  
worksheet.Range["A2"].CellStyle.FillPattern = ExcelPattern.Angle;
```

ASP.NET CORE

```
//Apply cell pattern  
worksheet.Range["A2"].CellStyle.FillPattern = ExcelPattern.Angle;
```

XAMARIN

```
//Apply cell pattern  
worksheet.Range["A2"].CellStyle.FillPattern = ExcelPattern.Angle;
```

Apply Border Settings

The XlsIO applies cell borders and format it through **IBorder** interface as shown as follows.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Apply borders
    worksheet.Range["A2"].CellStyle.Borders.LineStyle = ExcelLineStyle.Medium;
    worksheet.Range["A4"].CellStyle.Borders.LineStyle = ExcelLineStyle.Double;
    worksheet.Range["A6"].CellStyle.Borders.LineStyle = ExcelLineStyle.Dash_dot;
    worksheet.Range["A8"].CellStyle.Borders.LineStyle = ExcelLineStyle.Thick;
    worksheet.Range["C2"].CellStyle.Borders.LineStyle =
    ExcelLineStyle.Slanted_dash_dot;
    worksheet.Range["C4"].CellStyle.Borders.LineStyle = ExcelLineStyle.Hair;
    worksheet.Range["C6"].CellStyle.Borders.LineStyle =
    ExcelLineStyle.Medium_dash_dot_dot;
    worksheet.Range["C8"].CellStyle.Borders.LineStyle = ExcelLineStyle.Thin;
    //Apply Border using Border Index
    //Top Border
    worksheet.Range["E2"].CellStyle.Borders[ExcelBordersIndex.EdgeTop].LineStyle
    = ExcelLineStyle.Medium;
    //Left Border
    worksheet.Range["E4"].CellStyle.Borders[ExcelBordersIndex.EdgeLeft].LineStyl
    e = ExcelLineStyle.Double;
    //Bottom Border
    worksheet.Range["E6"].CellStyle.Borders[ExcelBordersIndex.EdgeBottom].LineSt
    yle = ExcelLineStyle.Dashed;
    //Right Border
    worksheet.Range["E8"].CellStyle.Borders[ExcelBordersIndex.EdgeRight].LineSty
    le = ExcelLineStyle.Thick;
    //DiagonalUp Border
    worksheet.Range["E10"].CellStyle.Borders[ExcelBordersIndex.DiagonalUp].LineS
    tyle = ExcelLineStyle.Thin;
    //DiagonalDown Border
    worksheet.Range["E12"].CellStyle.Borders[ExcelBordersIndex.DiagonalDown].Lin
    eStyle = ExcelLineStyle.Dotted;
    //Apply border color
    worksheet.Range["A2"].CellStyle.Borders.Color = ExcelKnownColors.Blue;
    //Setting the Border as Range
    worksheet.Range["G2:I8"].BorderAround();
    worksheet.Range["G2:I8"].BorderInside(ExcelLineStyle.Dash_dot,
    ExcelKnownColors.Red);
    workbook.SaveAs("BorderSettings.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
```

```

Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Apply borders
worksheet.Range("A2").CellStyle.Borders.LineStyle = ExcelLineStyle.Medium
worksheet.Range("A4").CellStyle.Borders.LineStyle = ExcelLineStyle.Double
worksheet.Range("A6").CellStyle.Borders.LineStyle = ExcelLineStyle.Dash_dot
worksheet.Range("A8").CellStyle.Borders.LineStyle = ExcelLineStyle.Thick
worksheet.Range("C2").CellStyle.Borders.LineStyle =
ExcelLineStyle.Slanted_dash_dot
worksheet.Range("C4").CellStyle.Borders.LineStyle = ExcelLineStyle.Hair
worksheet.Range("C6").CellStyle.Borders.LineStyle =
ExcelLineStyle.Medium_dash_dot_dot
worksheet.Range("C8").CellStyle.Borders.LineStyle = ExcelLineStyle.Thin
'Apply Border using Border Index
'Top Border
worksheet.Range("E2").CellStyle.Borders(ExcelBordersIndex.EdgeTop).LineStyle
= ExcelLineStyle.Medium
'Left Border
worksheet.Range("E4").CellStyle.Borders(ExcelBordersIndex.EdgeLeft).LineStyle
= ExcelLineStyle.Double
'Bottom Border
worksheet.Range("E6").CellStyle.Borders(ExcelBordersIndex.EdgeBottom).LineStyle
= ExcelLineStyle.Dashed
'Right Border
worksheet.Range("E8").CellStyle.Borders(ExcelBordersIndex.EdgeRight).LineStyle
= ExcelLineStyle.Thick
'DiagonalUp Border
worksheet.Range("E10").CellStyle.Borders(ExcelBordersIndex.DiagonalUp).LineStyle
= ExcelLineStyle.Thin
'DiagonalDown Border
worksheet.Range("E12").CellStyle.Borders(ExcelBordersIndex.DiagonalDown).LineStyle
= ExcelLineStyle.Dotted
'Apply border color
worksheet.Range("A2").CellStyle.Borders.Color = ExcelKnownColors.Blue
'Setting the Border as Range
worksheet.Range("G2:I8").BorderAround()
worksheet.Range("G2:I8").BorderInside(ExcelLineStyle.Dash_dot,
ExcelKnownColors.Red)
workbook.SaveAs("BorderSettings.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Apply borders
    worksheet.Range["A2"].CellStyle.Borders.LineStyle = ExcelLineStyle.Medium;
    worksheet.Range["A4"].CellStyle.Borders.LineStyle = ExcelLineStyle.Double;
    worksheet.Range["A6"].CellStyle.Borders.LineStyle = ExcelLineStyle.Dash_dot;
    worksheet.Range["A8"].CellStyle.Borders.LineStyle = ExcelLineStyle.Thick;
    worksheet.Range["C2"].CellStyle.Borders.LineStyle =
    ExcelLineStyle.Slanted_dash_dot;
    worksheet.Range["C4"].CellStyle.Borders.LineStyle = ExcelLineStyle.Hair;

```

```

worksheet.Range["C6"].CellStyle.Borders.LineStyle =
ExcelLineStyle.Medium_dash_dot_dot;
worksheet.Range["C8"].CellStyle.Borders.LineStyle = ExcelLineStyle.Thin;
//Apply Border using Border Index
//Top Border
worksheet.Range["E2"].CellStyle.Borders[ExcelBordersIndex.EdgeTop].LineStyle
= ExcelLineStyle.Medium;
//Left Border
worksheet.Range["E4"].CellStyle.Borders[ExcelBordersIndex.EdgeLeft].LineStyle
= ExcelLineStyle.Double;
//Bottom Border
worksheet.Range["E6"].CellStyle.Borders[ExcelBordersIndex.EdgeBottom].LineStyle
= ExcelLineStyle.Dashed;
//Right Border
worksheet.Range["E8"].CellStyle.Borders[ExcelBordersIndex.EdgeRight].LineStyle
= ExcelLineStyle.Thick;
//DiagonalUp Border
worksheet.Range["E10"].CellStyle.Borders[ExcelBordersIndex.DiagonalUp].LineStyle
= ExcelLineStyle.Thin;
//DiagonalDown Border
worksheet.Range["E12"].CellStyle.Borders[ExcelBordersIndex.DiagonalDown].LineStyle
= ExcelLineStyle.Dotted;
//Apply border color
worksheet.Range["A2"].CellStyle.Borders.Color = ExcelKnownColors.Blue;
//Setting the Border as Range
worksheet.Range["G2:I8"].BorderAround();
worksheet.Range["G2:I8"].BorderInside(ExcelLineStyle.Dash_dot,
ExcelKnownColors.Red);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "BorderSettings";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Apply borders
    worksheet.Range["A2"].CellStyle.Borders.LineStyle = ExcelLineStyle.Medium;
    worksheet.Range["A4"].CellStyle.Borders.LineStyle = ExcelLineStyle.Double;
    worksheet.Range["A6"].CellStyle.Borders.LineStyle = ExcelLineStyle.Dash_dot;
    worksheet.Range["A8"].CellStyle.Borders.LineStyle = ExcelLineStyle.Thick;
    worksheet.Range["C2"].CellStyle.Borders.LineStyle =
ExcelLineStyle.Slanted_dash_dot;
    worksheet.Range["C4"].CellStyle.Borders.LineStyle = ExcelLineStyle.Hair;
}

```

```

worksheet.Range["C6"].CellStyle.Borders.LineStyle =
ExcelLineStyle.Medium_dash_dot_dot;
worksheet.Range["C8"].CellStyle.Borders.LineStyle = ExcelLineStyle.Thin;
//Apply Border using Border Index
//Top Border
worksheet.Range["E2"].CellStyle.Borders[ExcelBordersIndex.EdgeTop].LineStyle
= ExcelLineStyle.Medium;
//Left Border
worksheet.Range["E4"].CellStyle.Borders[ExcelBordersIndex.EdgeLeft].LineStyle
= ExcelLineStyle.Double;
//Bottom Border
worksheet.Range["E6"].CellStyle.Borders[ExcelBordersIndex.EdgeBottom].LineStyle
= ExcelLineStyle.Dashed;
//Right Border
worksheet.Range["E8"].CellStyle.Borders[ExcelBordersIndex.EdgeRight].LineStyle
= ExcelLineStyle.Thick;
//DiagonalUp Border
worksheet.Range["E10"].CellStyle.Borders[ExcelBordersIndex.DiagonalUp].LineStyle
= ExcelLineStyle.Thin;
//DiagonalDown Border
worksheet.Range["E12"].CellStyle.Borders[ExcelBordersIndex.DiagonalDown].LineStyle
= ExcelLineStyle.Dotted;
//Apply border color
worksheet.Range["A2"].CellStyle.Borders.Color = ExcelKnownColors.Blue;
//Setting the Border as Range
worksheet.Range["G2:I8"].BorderAround();
worksheet.Range["G2:I8"].BorderInside(ExcelLineStyle.Dash_dot,
ExcelKnownColors.Red);
//Saving the workbook as stream
FileStream stream = new FileStream("BorderSettings.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

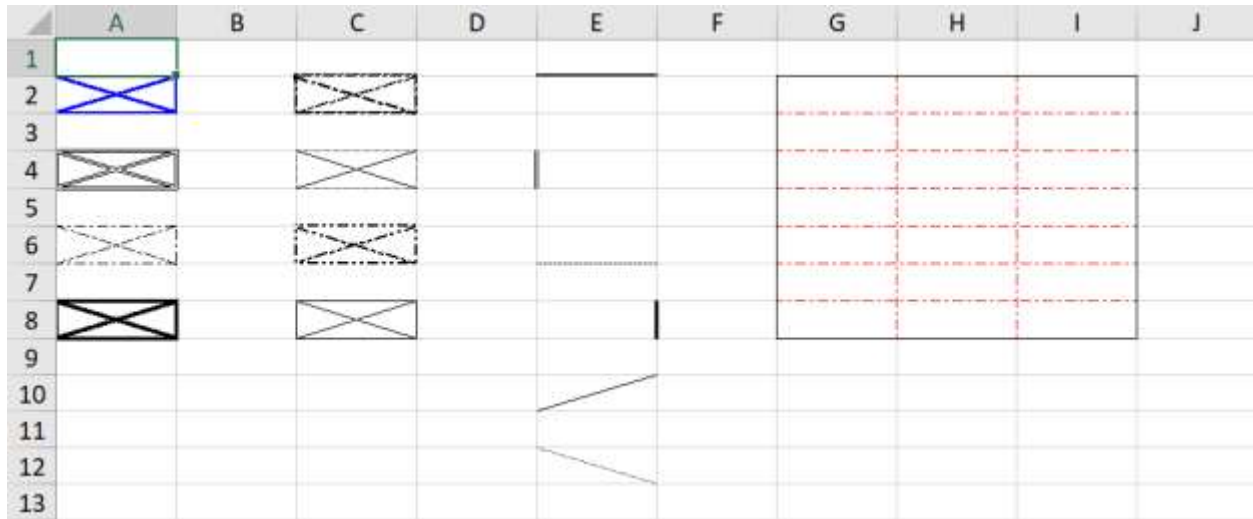
```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Apply borders
    worksheet.Range["A2"].CellStyle.Borders.LineStyle = ExcelLineStyle.Medium;
    worksheet.Range["A4"].CellStyle.Borders.LineStyle = ExcelLineStyle.Double;
    worksheet.Range["A6"].CellStyle.Borders.LineStyle = ExcelLineStyle.Dash_dot;
    worksheet.Range["A8"].CellStyle.Borders.LineStyle = ExcelLineStyle.Thick;
    worksheet.Range["C2"].CellStyle.Borders.LineStyle =
ExcelLineStyle.Slanted_dash_dot;
    worksheet.Range["C4"].CellStyle.Borders.LineStyle = ExcelLineStyle.Hair;
    worksheet.Range["C6"].CellStyle.Borders.LineStyle =
ExcelLineStyle.Medium_dash_dot_dot;
    worksheet.Range["C8"].CellStyle.Borders.LineStyle = ExcelLineStyle.Thin;
    //Apply Border using Border Index
    //Top Border

```

```
worksheet.Range["E2"].CellStyle.Borders[ExcelBordersIndex.EdgeTop].LineStyle
= ExcelLineStyle.Medium;
//Left Border
worksheet.Range["E4"].CellStyle.Borders[ExcelBordersIndex.EdgeLeft].LineStyl
e = ExcelLineStyle.Double;
//Bottom Border
worksheet.Range["E6"].CellStyle.Borders[ExcelBordersIndex.EdgeBottom].LineSt
yle = ExcelLineStyle.Dashed;
//Right Border
worksheet.Range["E8"].CellStyle.Borders[ExcelBordersIndex.EdgeRight].LineStyl
e = ExcelLineStyle.Thick;
//DiagonalUp Border
worksheet.Range["E10"].CellStyle.Borders[ExcelBordersIndex.DiagonalUp].Lines
tyle = ExcelLineStyle.Thin;
//DiagonalDown Border
worksheet.Range["E12"].CellStyle.Borders[ExcelBordersIndex.DiagonalDown].Lin
estyle = ExcelLineStyle.Dotted;
//Apply border color
worksheet.Range["A2"].CellStyle.Borders.Color = ExcelKnownColors.Blue;
//Setting the Border as Range
worksheet.Range["G2:I8"].BorderAround();
worksheet.Range["G2:I8"].BorderInside(ExcelLineStyle.Dash_dot,
ExcelKnownColors.Red);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Border
Settings.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("BorderSettings.xls
x", "application/msexcel", stream);
}
}
```

The output of the previous code is shown in the following screenshot:



HTML String Formatting

HTML string generates a string of characters with different formatting styles using different HTML tags for each character. This makes it easy to manipulate the text or value in the worksheet range as each character is independent and doesn't depend on hierarchical tag structure.

XlsIO supports adding HTML Rich-Text to a range of cells in worksheet.

Applying HTML String

The following code snippet illustrates how to read and write HTML Rich-Text using `HtmlString` property of `IRange`.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(3);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Add HTML string
    worksheet.Range["A1"].HtmlString = "<font style=\"color:red;font-family:Magneto;font-size:12px; \">Welcome Syncfusion</font>";
    //Assign HTML string as text to different range
    worksheet.Range["A2"].Text = worksheet.Range["A1"].HtmlString;
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(3)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Add HTML string
worksheet.Range("A1").HtmlString = "<font style=\"color:red;font-family:Magneto;font-size:12px; \">Welcome Syncfusion</font>"
'Assign HTML string as text to different range
```

```
worksheet.Range("A2").Text = worksheet.Range("A1").HtmlString
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
//XlsIO supports adding HTML string to worksheet range in Windows Forms,  
WPF, ASP.NET, ASP.NET MVC platforms alone.
```

ASP.NET CORE

```
//XlsIO supports adding HTML string to worksheet range in Windows Forms,  
WPF, ASP.NET, ASP.NET MVC platforms alone.
```

XAMARIN

```
//XlsIO supports adding HTML string to worksheet range in Windows Forms,  
WPF, ASP.NET, ASP.NET MVC platforms alone.
```

Supported Tags

The following are the list of tags supported in addition to HTML string.

- \
 - Defines a paragraph
- \ - Defines font, color and size of text
- \&.h6> - Defines HTML headings &.h6>
- \ - Defines a hyperlink
- \ - Defines italic text
- \ - Underlines the text
- \ - Defines bold text
- \ - Defines subscript
- \ - Defines superscript
- \
 - Inserts link break
- \ - Strikes out the text
- \ - Defines important text

These mentioned tags do support the following style attributes.

- Color
- Font-family
- Text-decoration
- Font-size
- Size
- Face

Rich-Text Formatting

You can format each character in a cell with different font styles. XlsIO reads and writes rich-text by using the **IRichTextString** interface.

Note: Currently XlsIO cannot process and write RTF codes to cells.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Add Text
    IRange range = worksheet.Range["A1"];
    range.Text = "RichText";
    IRichTextString richText = range.RichText;
    //Formatting first 4 characters
    IFont redFont = workbook.CreateFont();
    redFont.Bold = true;
    redFont.Italic = true;
    redFont.RGBColor = Color.Red;
    richText.SetFont(0, 3, redFont);
    //Formatting last 4 characters
    IFont blueFont = workbook.CreateFont();
    blueFont.Bold = true;
    blueFont.Italic = true;
    blueFont.RGBColor = Color.Blue;
    richText.SetFont(4, 7, blueFont);
    workbook.SaveAs("RichText.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Add Text
Dim range As IRange = worksheet.Range("A1")
range.Text = "RichText"
Dim richText As IRichTextString = range.RichText
'Formatting first 4 characters
Dim redFont As IFont = workbook.CreateFont()
redFont.Bold = True
redFont.Italic = True
redFont.RGBColor = Color.Red
richText.SetFont(0, 3, redFont)
'Formatting last 4 characters
Dim blueFont As IFont = workbook.CreateFont()
blueFont.Bold = True
blueFont.Italic = True
blueFont.RGBColor = Color.Blue
richText.SetFont(4, 7, blueFont)
```

```
workbook.SaveAs("RichText.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Add Text
    IRange range = worksheet.Range["A1"];
    range.Text = "RichText";
    IRichTextString richText = range.RichText;
    //Formatting first 4 characters.
    IFont redFont = workbook.CreateFont();
    redFont.Bold = true;
    redFont.Italic = true;
    redFont.RGBColor = Color.FromArgb(255, 255, 0, 0);
    richText.SetFont(0, 3, redFont);
    //Formatting last 4 characters.
    IFont blueFont = workbook.CreateFont();
    blueFont.Bold = true;
    blueFont.Italic = true;
    blueFont.RGBColor = Color.FromArgb(255, 0, 0, 255);
    richText.SetFont(4, 7, blueFont);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "RichText";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Add Text
    IRange range = worksheet.Range["A1"];
    range.Text = "RichText";
    IRichTextString richText = range.RichText;
    //Formatting first 4 characters.
    IFont redFont = workbook.CreateFont();
    redFont.Bold = true;
    redFont.Italic = true;
    redFont.RGBColor = Color.Red;
```

```

richText.SetFont(0, 3, redFont);
//Formatting last 4 characters.
IFont blueFont = workbook.CreateFont();
blueFont.Bold = true;
blueFont.Italic = true;
blueFont.RGBColor = Color.Blue;
richText.SetFont(4, 7, blueFont);
//Saving the workbook as stream
FileStream stream = new FileStream("RichText.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Add Text
    IRange range = worksheet.Range["A1"];
    range.Text = "RichText";
    IRichTextString richText = range.RichText;
    //Formatting first 4 characters.
    IFont redFont = workbook.CreateFont();
    redFont.Bold = true;
    redFont.Italic = true;
    redFont.RGBColor = Syncfusion.Drawing.Color.Red;
    richText.SetFont(0, 3, redFont);
    //Formatting last 4 characters.
    IFont blueFont = workbook.CreateFont();
    blueFont.Bold = true;
    blueFont.Italic = true;
    blueFont.RGBColor = Syncfusion.Drawing.Color.Blue;
    richText.SetFont(4, 7, blueFont);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Refer to the xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("RichText.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("RichText.xlsx", "application/msexcel", stream);
    }
}

```

```
}
}
```

The output of the previous code is shown as follows:

	A	B
1	<i>Rich Text</i>	
2		

Working with Data

Importing Data to Worksheets

XlsIO provides the ability to import data into a worksheet from the following data.

- Data Table
- Data Column
- Data View
- Collection Objects
- Nested Collection Objects
- Array

Import Data from DataTable

The following code snippet illustrates on how to import a DataTable into a worksheet using **ImportDataTable** method.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Initialize the DataTable
    DataTable table = SampleDataTable();
    //Import DataTable to the worksheet.
    worksheet.ImportDataTable(table, true, 1, 1);
    workbook.SaveAs("ImportFromDT.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Initialize the DataTable
Dim table As DataTable = sampleDataTable()
'Import DataTable to the worksheet
worksheet.ImportDataTable(table, True, 1, 1)
workbook.SaveAs("ImportFromDT.xlsx")
```

End Using**UWP**

//XlsIO supports importing of data from data table to worksheet in Windows Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Core (2.0 onwards) platforms alone.

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Initialize the DataTable
    DataTable table = SampleDataTable();
    //Import DataTable to the worksheet
    worksheet.ImportDataTable(table, true, 1, 1);
    //Saving the workbook as stream
    FileStream stream = new FileStream("ImportFromDT.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

//XlsIO supports importing of data from data table to worksheet in Windows Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Core (2.0 onwards) platforms alone.

Import Data from DataColumn

The following code snippet illustrates how to import DataColumn into a worksheet using **ImportDataColumn** method.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Initialize the DataTable
    DataTable table = SampleDataTable();
    //Import Data Column to the worksheet
    DataColumn column = table.Columns[0];
    worksheet.ImportDataColumn(column, true, 1, 1);
    workbook.SaveAs("ImportFromDT.xlsx");
}
```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Initialize the DataTable
Dim table As DataTable = sampleDataTable()
'Import DataColumn to the worksheet
Dim column As DataColumn = table.Columns(0)
worksheet.ImportDataColumn(column, True, 1, 1)
workbook.SaveAs("ImportFromDT.xlsx")
End Using

```

UWP

//XlsIO supports importing data column to worksheet in Windows Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Core (2.0 onwards) platforms alone.

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Initialize the DataTable
    DataTable table = SampleDataTable();
    //Import Data Column to the worksheet
    DataColumn column = table.Columns[0];
    worksheet.ImportDataColumn(column, true, 1, 1);
    //Saving the workbook as stream
    FileStream stream = new FileStream("ImportFromDT.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

//XlsIO supports importing data column to worksheet in Windows Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Core (2.0 onwards) platforms alone.

Import Data from DataView

The following code snippet illustrates how to import DataView into a worksheet using **ImportDataView** method.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);

```

```

IWorksheet worksheet = workbook.Worksheets[0];
//Initialize the DataTable
DataTable table = SampleDataTable();
//Import DataView to the worksheet
DataView view = table.DefaultView;
worksheet.ImportDataView(view, true, 1, 1);
workbook.SaveAs("ImportFromDT.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Initialize the DataTable
Dim table As DataTable = sampleDataTable()
'Import DataView to the worksheet
Dim view As DataView = table.DefaultView
worksheet.ImportDataView(view, True, 1, 1)
workbook.SaveAs("ImportFromDT.xlsx")
End Using

```

UWP

//XlsIO supports importing data view to worksheet in Windows Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Core (2.0 onwards) platforms alone.

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Initialize the DataTable
DataTable table = SampleDataTable();
//Import DataView to the worksheet
DataView view = table.DefaultView;
worksheet.ImportDataView(view, true, 1, 1);
//Saving the workbook as stream
FileStream stream = new FileStream("ImportFromDT.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

//XlsIO supports importing data view to worksheet in Windows Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Core (2.0 onwards) platforms alone.

Import Data from Collection Objects

Essential XlsIO allows you to import data directly from Collection Objects as shown below.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Import the data to worksheet
    IList<Customer> reports = GetSalesReports();
    worksheet.ImportData(reports, 2, 1, false);
    workbook.SaveAs("ImportFromDT.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Import the data to worksheet
Dim reports As IList(Of Customer) = GetSalesReports()
worksheet.ImportData(reports, 2, 1, False)
workbook.SaveAs("ImportFromDT.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Import the data to worksheet
    IList<Customer> reports = GetSalesReports();
    worksheet.ImportData(reports, 2, 1, false);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "ImportFromDT";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx" });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE


```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Import the data to worksheet
    IList<Customer> reports = GetSalesReports();
    worksheet.ImportData(reports, 2, 1, false);
    //Saving the workbook as stream
    FileStream stream = new FileStream("ImportFromDT.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Import the data to worksheet
    IList<Customer> reports = GetSalesReports();
    worksheet.ImportData(reports, 2, 1, false);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Import
        FromDT.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ImportFromDT.xlsx"
        , "application/msexcel", stream);
    }
}

```

The following code snippet provides supporting class for the above code. Here, the attributes `DisplayNameAttribute` and `Bindable` are used.

- [DisplayNameAttribute](#) - to customize the column header name while importing.
- [BindableAttribute](#) - to skip a property while importing.

C#

```

//Gets a list of sales reports
public static List<Customer> GetSalesReports()
{
    List<Customer> reports = new List<Customer>();
    reports.Add(new Customer("Andy Bernard", "45000", "58000"));
    reports.Add(new Customer("Jim Halpert", "34000", "65000"));
    reports.Add(new Customer("Karen Fillippelli", "75000", "64000"));
    reports.Add(new Customer("Phyllis Lapin", "56500", "33600"));
    reports.Add(new Customer("Stanley Hudson", "46500", "52000"));
    return reports;
}

//Customer details
public class Customer
{
    [DisplayNameAttribute("Sales Person Name")]
    public string SalesPerson { get; set; }
    [Bindable(false)]
    public string SalesJanJun { get; set; }
    public string SalesJulDec { get; set; }
    public Customer(string name, string janToJun, string julToDec)
    {
        SalesPerson = name;
        SalesJanJun = janToJun;
        SalesJulDec = julToDec;
    }
}

```

VB.NET

```

'Gets a list of sales reports
Public Function GetSalesReports() As List(Of Customer)
Dim reports As New List(Of Customer)()
reports.Add(New Customer("Andy Bernard", "45000", "58000"))
reports.Add(New Customer("Jim Halpert", "34000", "65000"))
reports.Add(New Customer("Karen Fillippelli", "75000", "64000"))
reports.Add(New Customer("Phyllis Lapin", "56500", "33600"))
reports.Add(New Customer("Stanley Hudson", "46500", "52000"))
Return reports
End Function

'Customer details
Public Class Customer
Private m_SalesPerson As String
Private m_SalesJanJun As String
Private m_SalesJulDec As String
<DisplayNameAttribute("Sales Person Name")>
Public Property SalesPerson() As String
Get
Return m_SalesPerson
End Get
Set(value As String)
m_SalesPerson = Value
End Set
End Property
<Bindable(False)>

```

```

Public Property SalesJanJun() As String
Get
Return m_SalesJanJun
End Get
Set(value As String)
m_SalesJanJun = Value
End Set
End Property
Public Property SalesJulDec() As String
Get
Return m_SalesJulDec
End Get
Set(value As String)
m_SalesJulDec = Value
End Set
End Property
Public Sub New(name As String, janToJun As String, julToDec As String)
SalesPerson = name
SalesJanJun = janToJun
SalesJulDec = julToDec
End Sub
End Class

```

UWP

```

//Gets a list of sales reports
public static List<Customer> GetSalesReports()
{
List<Customer> reports = new List<Customer>();
reports.Add(new Customer("Andy Bernard", "45000", "58000"));
reports.Add(new Customer("Jim Halpert", "34000", "65000"));
reports.Add(new Customer("Karen Fillippelli", "75000", "64000"));
reports.Add(new Customer("Phyllis Lapin", "56500", "33600" ));
reports.Add(new Customer("Stanley Hudson", "46500", "52000"));
return reports;
}
//Customer details
public class Customer
{
[DisplayNameAttribute("Sales Person Name")]
public string SalesPerson { get; set; }
[Bindable(false)]
public string SalesJanJun { get; set; }
public string SalesJulDec { get; set; }
public Customer(string name, string janToJun, string julToDec)
{
SalesPerson = name;
SalesJanJun = janToJun;
SalesJulDec = julToDec;
}
}
}

```

ASP.NET CORE

```

//Gets a list of sales reports
public static List<Customer> GetSalesReports()

```

```

{
    List<Customer> reports = new List<Customer>();
    reports.Add(new Customer("Andy Bernard", "45000", "58000"));
    reports.Add(new Customer("Jim Halpert", "34000", "65000"));
    reports.Add(new Customer("Karen Fillippelli", "75000", "64000"));
    reports.Add(new Customer("Phyllis Lapin", "56500", "33600" ));
    reports.Add(new Customer("Stanley Hudson", "46500", "52000"));
    return reports;
}
//Customer details
public class Customer
{
    [DisplayNameAttribute("Sales Person Name")]
    public string SalesPerson { get; set; }
    [Bindable(false)]
    public string SalesJanJun { get; set; }
    public string SalesJulDec { get; set; }
    public Customer(string name, string janToJun, string julToDec)
    {
        SalesPerson = name;
        SalesJanJun = janToJun;
        SalesJulDec = julToDec;
    }
}

```

XAMARIN

```

//Gets a list of sales reports
public static List<Customer> GetSalesReports()
{
    List<Customer> reports = new List<Customer>();
    reports.Add(new Customer("Andy Bernard", "45000", "58000"));
    reports.Add(new Customer("Jim Halpert", "34000", "65000"));
    reports.Add(new Customer("Karen Fillippelli", "75000", "64000"));
    reports.Add(new Customer("Phyllis Lapin", "56500", "33600" ));
    reports.Add(new Customer("Stanley Hudson", "46500", "52000"));
    return reports;
}
//Customer details
public class Customer
{
    [DisplayNameAttribute("Sales Person Name")]
    public string SalesPerson { get; set; }
    [Bindable(false)]
    public string SalesJanJun { get; set; }
    public string SalesJulDec { get; set; }
    public Customer(string name, string janToJun, string julToDec)
    {
        SalesPerson = name;
        SalesJanJun = janToJun;
        SalesJulDec = julToDec;
    }
}

```

Import Data Options

ExcelImportDataOptions is a support class for ImportData() method which contains various properties to import data with formatting.

ExcelImportDataOptions class contains the following properties:

FirstRow - Specifies first row from where the data should be imported.

FirstColumn - Specifies first column from where the data should be imported.

IncludeHeader - Specifies whether class properties names must be imported or not.

PreserveTypes - Indicates whether XlsIO should preserve column types from Data. By default, preserve type is TRUE. Setting it to True will import data based on column type, otherwise will import based on value type.

The following code snippet illustrates how to import collection objects into a worksheet using ImportData method with ExcelImportDataOptions class.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Import the data to worksheet with Import Data Options
    IList<Customer> reports = GetSalesReports();
    ExcelImportDataOptions importDataOptions = new ExcelImportDataOptions();
    importDataOptions.FirstRow = 2;
    importDataOptions.FirstColumn = 1;
    importDataOptions.IncludeHeader = false;
    importDataOptions.PreserveTypes = false;
    worksheet.ImportData(reports, importDataOptions);
    workbook.SaveAs("ImportData.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Import the data to worksheet with Import Data Options
Dim reports As IList(Of Customer) = GetSalesReports()
Dim importDataOptions As ExcelImportDataOptions = New
ExcelImportDataOptions()
importDataOptions.FirstRow = 2
importDataOptions.FirstColumn = 1
importDataOptions.IncludeHeader = False
importDataOptions.PreserveTypes = False
worksheet.ImportData(output, importDataOptions)
workbook.SaveAs("ImportData.xlsx")
End Using
```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Import the data to worksheet with Import Data Options
    IList<Customer> reports = GetSalesReports();
    ExcelImportDataOptions importDataOptions = new ExcelImportDataOptions();
    importDataOptions.FirstRow = 2;
    importDataOptions.FirstColumn = 1;
    importDataOptions.IncludeHeader = false;
    importDataOptions.PreserveTypes = false;
    worksheet.ImportData(reports, importDataOptions);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "ImportData";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Import the data to worksheet with Import Data Options
    IList<Customer> reports = GetSalesReports();
    ExcelImportDataOptions importDataOptions = new ExcelImportDataOptions();
    importDataOptions.FirstRow = 2;
    importDataOptions.FirstColumn = 1;
    importDataOptions.IncludeHeader = false;
    importDataOptions.PreserveTypes = false;
    worksheet.ImportData(reports, importDataOptions);
    //Saving the workbook as stream
    FileStream stream = new FileStream("ImportData.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;

```

```

application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Import the data to worksheet with Import Data Options
IList<Customer> reports = GetSalesReports();
ExcelImportDataOptions importDataOptions = new ExcelImportDataOptions();
importDataOptions.FirstRow = 2;
importDataOptions.FirstColumn = 1;
importDataOptions.IncludeHeader = false;
importDataOptions.PreserveTypes = false;
worksheet.ImportData(reports, importDataOptions);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Import
Data.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ImportData.xlsx",
"application/msexcel", stream);
}
}

```

The following code snippet provides supporting class for the above code.

C#

```

//Gets a list of sales reports
public static List<Customer> GetSalesReports()
{
List<Customer> reports = new List<Customer>();
reports.Add(new Customer("Andy Bernard", "45000", "58000"));
reports.Add(new Customer("Jim Halpert", "34000", "65000"));
reports.Add(new Customer("Karen Fillippelli", "75000", "64000"));
reports.Add(new Customer("Phyllis Lapin", "56500", "33600"));
reports.Add(new Customer("Stanley Hudson", "46500", "52000"));
return reports;
}
//Customer details
public class Customer
{
public string SalesPerson { get; set; }
public string SalesJanJun { get; set; }
public string SalesJulDec { get; set; }
public Customer(string name, string janToJun, string julToDec)
{
SalesPerson = name;

```

```
SalesJanJun = janToJun;
SalesJulDec = julToDec;
}
}
```

VB.NET

```
'Gets a list of sales reports
Public Function GetSalesReports() As List(Of Customer)
Dim reports As New List(Of Customer) ()
reports.Add(New Customer("Andy Bernard", "45000", "58000"))
reports.Add(New Customer("Jim Halpert", "34000", "65000"))
reports.Add(New Customer("Karen Fillippelli", "75000", "64000"))
reports.Add(New Customer("Phyllis Lapin", "56500", "33600"))
reports.Add(New Customer("Stanley Hudson", "46500", "52000"))
Return reports
End Function

'Customer details
Public Class Customer
Private m_SalesPerson As String
Private m_SalesJanJun As String
Private m_SalesJulDec As String
Public Property SalesPerson() As String
Get
Return m_SalesPerson
End Get
Set(value As String)
m_SalesPerson = Value
End Set
End Property
Public Property SalesJanJun() As String
Get
Return m_SalesJanJun
End Get
Set(value As String)
m_SalesJanJun = Value
End Set
End Property
Public Property SalesJulDec() As String
Get
Return m_SalesJulDec
End Get
Set(value As String)
m_SalesJulDec = Value
End Set
End Property
Public Sub New(name As String, janToJun As String, julToDec As String)
SalesPerson = name
SalesJanJun = janToJun
SalesJulDec = julToDec
End Sub
End Class
```

UWP

```
//Gets a list of sales reports
```



```

public static List<Customer> GetSalesReports()
{
    List<Customer> reports = new List<Customer>();
    reports.Add(new Customer("Andy Bernard", "45000", "58000"));
    reports.Add(new Customer("Jim Halpert", "34000", "65000"));
    reports.Add(new Customer("Karen Fillippelli", "75000", "64000"));
    reports.Add(new Customer("Phyllis Lapin", "56500", "33600" ));
    reports.Add(new Customer("Stanley Hudson", "46500", "52000"));
    return reports;
}
//Customer details
public class Customer
{
    public string SalesPerson { get; set; }
    public string SalesJanJun { get; set; }
    public string SalesJulDec { get; set; }
    public Customer(string name, string janToJun, string julToDec)
    {
        SalesPerson = name;
        SalesJanJun = janToJun;
        SalesJulDec = julToDec;
    }
}

```

ASP.NET CORE

```

//Gets a list of sales reports
public static List<Customer> GetSalesReports()
{
    List<Customer> reports = new List<Customer>();
    reports.Add(new Customer("Andy Bernard", "45000", "58000"));
    reports.Add(new Customer("Jim Halpert", "34000", "65000"));
    reports.Add(new Customer("Karen Fillippelli", "75000", "64000"));
    reports.Add(new Customer("Phyllis Lapin", "56500", "33600" ));
    reports.Add(new Customer("Stanley Hudson", "46500", "52000"));
    return reports;
}
//Customer details
public class Customer
{
    public string SalesPerson { get; set; }
    public string SalesJanJun { get; set; }
    public string SalesJulDec { get; set; }
    public Customer(string name, string janToJun, string julToDec)
    {
        SalesPerson = name;
        SalesJanJun = janToJun;
        SalesJulDec = julToDec;
    }
}

```

XAMARIN

```

//Gets a list of sales reports
public static List<Customer> GetSalesReports()
{

```

```

List<Customer> reports = new List<Customer>();
reports.Add(new Customer("Andy Bernard", "45000", "58000"));
reports.Add(new Customer("Jim Halpert", "34000", "65000"));
reports.Add(new Customer("Karen Fillippelli", "75000", "64000"));
reports.Add(new Customer("Phyllis Lapin", "56500", "33600" ));
reports.Add(new Customer("Stanley Hudson", "46500", "52000"));
return reports;
}
//Customer details
public class Customer
{
public string SalesPerson { get; set; }
public string SalesJanJun { get; set; }
public string SalesJulDec { get; set; }
public Customer(string name, string janToJun, string julToDec)
{
SalesPerson = name;
SalesJanJun = janToJun;
SalesJulDec = julToDec;
}
}

```

Import Data from Nested Collection Objects

Import hierarchical data from nested collections to Excel worksheet helps the user to analyze data in its structure. XlsIO provides more flexible options to analyze such data by importing in different layouts and grouping the imported data.

Data import can be done with the layout options:

- **Default** - Parent records imported in the first row of its collection.
- **Merge** - Parent records imported in merged rows.
- **Repeat** - Parent records imported in all the rows.

Imported data can be grouped with the grouping options:

- **Expand** – Imported data will be grouped and expanded.
- **Collapse** – Imported data will be grouped and collapsed at first level, by default.

Let's see these options in detail along with code examples and screenshots.

Layout Options

Default layout option

This option adds the property value once per object for the corresponding records in the column while importing.

The following code snippet illustrates how to import data directly from nested collection objects with default layout option. The input XML file used in the code can be downloaded [here](#).

C#

```

using Syncfusion.XlsIO;
using System.Collections.Generic;
using System.ComponentModel;
using System.IO;

```

```

using System.Xml.Serialization;
namespace ImportFromNestedCollection
{
    class Program
    {
        static void Main(string[] args)
        {
            ImportData();
        }
        //Main method to import data from nested collection to Excel worksheet.
        private static void ImportData()
        {
            ExcelEngine excelEngine = new ExcelEngine();
            IApplication application = excelEngine.Excel;
            application.DefaultVersion = ExcelVersion.Excel2016;
            IWorkbook workbook = excelEngine.Excel.Workbooks.Create(1);
            IWorksheet worksheet = workbook.Worksheets[0];
            IList<Brand> vehicles = GetVehicleDetails();
            ExcelImportDataOptions importDataOptions = new ExcelImportDataOptions();
            //Imports from 4th row.
            importDataOptions.FirstRow = 4;
            //Imports column headers.
            importDataOptions.IncludeHeader = true;
            //Set layout options.
            importDataOptions.NestedDataLayoutOptions =
            ExcelNestedDataLayoutOptions.Default;
            //Import data from the nested collection.
            worksheet.ImportData(vehicles, importDataOptions);
            //Apply style to headers
            worksheet["A1:C2"].Merge();
            worksheet["A1"].Text = "Automobile Brands in the US";
            worksheet.UsedRange.AutofitColumns();
            workbook.SaveAs("ImportData.xlsx");
            workbook.Close();
            excelEngine.Dispose();
        }
        //Helper method to load data from XML file and add them in collections.
        private static IList<Brand> GetVehicleDetails()
        {
            XmlSerializer deserializer = new XmlSerializer(typeof(BrandObjects));
            //Read data from XML file.
            TextReader textReader = new StreamReader(@"..\..\Data\ExportData.xml");
            BrandObjects brands = (BrandObjects)deserializer.Deserialize(textReader);
            //Initialize parent collection to add data from XML file.
            List<Brand> list = new List<Brand>();
            string brandName = brands.BrandObject[0].BrandName;
            string vehicleType = brands.BrandObject[0].VahicleType;
            string modelName = brands.BrandObject[0].ModelName;
            //Parent class
            Brand brand = new Brand(brandName);
            brand.VehicleTypes = new List<VehicleType>();
            VehicleType vehicle = new VehicleType(vehicleType);
            vehicle.Models = new List<Model>();
            Model model = new Model(modelName);
            brand.VehicleTypes.Add(vehicle);
            list.Add(brand);
            foreach (BrandObject brandObj in brands.BrandObject)

```

```

{
    if (brandName == brandObj.BrandName)
    {
        if (vehicleType == brandObj.VehicleType)
        {
            vehicle.Models.Add(new Model(brandObj.ModelName));
            continue;
        }
        else
        {
            vehicle = new VehicleType(brandObj.VehicleType);
            vehicle.Models = new List<Model>();
            vehicle.Models.Add(new Model(brandObj.ModelName));
            brand.VehicleTypes.Add(vehicle);
            vehicleType = brandObj.VehicleType;
        }
        continue;
    }
    else
    {
        brand = new Brand(brandObj.BrandName);
        vehicle = new VehicleType(brandObj.VehicleType);
        vehicle.Models = new List<Model>();
        vehicle.Models.Add(new Model(brandObj.ModelName));
        brand.VehicleTypes = new List<VehicleType>();
        brand.VehicleTypes.Add(vehicle);
        vehicleType = brandObj.VehicleType;
        list.Add(brand);
        brandName = brandObj.BrandName;
    }
}
textReader.Close();
return list;
}
}

//Parent Class
public class Brand
{
    private string m_brandName;
    [DisplayNameAttribute("Brand")]
    public string BrandName
    {
        get { return m_brandName; }
        set { m_brandName = value; }
    }

    //Vehicle Types Collection
    private IList<VehicleType> m_vehicleTypes;
    public IList<VehicleType> VehicleTypes
    {
        get { return m_vehicleTypes; }
        set { m_vehicleTypes = value; }
    }
    public Brand(string brandName)
    {
        m_brandName = brandName;
    }
}

```

```

//Child Class
public class VehicleType
{
    private string m_vehicleName;
    [DisplayNameAttribute("Vehicle Type")]
    public string VehicleName
    {
        get { return m_vehicleName; }
        set { m_vehicleName = value; }
    }
    //Models collection
    private IList<Model> m_models;
    public IList<Model> Models
    {
        get { return m_models; }
        set { m_models = value; }
    }
    public VehicleType(string vehicle)
    {
        m_vehicleName = vehicle;
    }
}
//Sub-child Class
public class Model
{
    private string m_modelName;
    [DisplayNameAttribute("Model")]
    public string ModelName
    {
        get { return m_modelName; }
        set { m_modelName = value; }
    }
    public Model(string name)
    {
        m_modelName = name;
    }
}
//Helper Classes
[XmlRootAttribute("BrandObjects")]
public class BrandObjects
{
    [XmlElement("BrandObject")]
    public BrandObject[] BrandObject { get; set; }
}
public class BrandObject
{
    public string BrandName { get; set; }
    public string VehicleType { get; set; }
    public string ModelName { get; set; }
}
}

```

VB.NET

```

Imports Syncfusion.XlsIO
Imports System.Collections.Generic

```

```

Imports System.ComponentModel
Imports System.IO
Imports System.Xml.Serialization
Namespace ImportFromNestedCollection
Class Program
Private Shared Sub Main(ByVal args As String())
ImportData()
End Sub
'Main method to import data from nested collection to Excel worksheet.
Private Shared Sub ImportData()
Dim excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim vehicles As IList(Of Brand) = GetVehicleDetails()
Dim importDataOptions As ExcelImportDataOptions = New
ExcelImportDataOptions()
'Imports from 4th row.
importDataOptions.FirstRow = 4
'Imports column headers.
importDataOptions.IncludeHeader = True
'Set layout options.
importDataOptions.NestedDataLayoutOptions =
ExcelNestedDataLayoutOptions.Default
'Import data from the nested collection.
worksheet.ImportData(vehicles, importDataOptions)
'Apply style to headers
worksheet("A1:C2").Merge()
worksheet("A1").Text = "Automobile Brands in the US"
worksheet.UsedRange.AutofitColumns()
workbook.SaveAs("ImportData.xlsx")
workbook.Close()
excelEngine.Dispose()
End Sub
'Helper method to load data from XML file and add them in collections.
Private Shared Function GetVehicleDetails() As IList(Of Brand)
Dim deserializer As XmlSerializer = New XmlSerializer(GetType(BrandObjects))
'Read data from XML file.
Dim textReader As TextReader = New StreamReader("../Data/ExportData.xml")
Dim brands As BrandObjects = CType(deserializer.Deserialize(textReader),
BrandObjects)
'Initialize parent collection to add data from XML file.
Dim list As List(Of Brand) = New List(Of Brand)()
Dim brandName As String = brands.BrandObject(0).BrandName
Dim vehicleType As String = brands.BrandObject(0).VehicleType
Dim modelName As String = brands.BrandObject(0).ModelName
'Parent class
Dim brand As Brand = New Brand(brandName)
brand.VehicleTypes = New List(Of VehicleType)()
Dim vehicle As VehicleType = New VehicleType(vehicleType)
vehicle.Models = New List(Of Model)()
Dim model As Model = New Model(modelName)
brand.VehicleTypes.Add(vehicle)
list.Add(brand)
For Each brandObj As BrandObject In brands.BrandObject
If brandName = brandObj.BrandName Then

```

```

If vehicleType = brandObj.VehicleType Then
vehicle.Models.Add(New Model(brandObj.ModelName))
Continue For
Else
vehicle = New VehicleType(brandObj.VehicleType)
vehicle.Models = New List(Of Model)()
vehicle.Models.Add(New Model(brandObj.ModelName))
brand.VehicleTypes.Add(vehicle)
vehicleType = brandObj.VehicleType
End If
Continue For
Else
brand = New Brand(brandObj.BrandName)
vehicle = New VehicleType(brandObj.VehicleType)
vehicle.Models = New List(Of Model)()
vehicle.Models.Add(New Model(brandObj.ModelName))
brand.VehicleTypes = New List(Of VehicleType)()
brand.VehicleTypes.Add(vehicle)
vehicleType = brandObj.VehicleType
list.Add(brand)
brandName = brandObj.BrandName
End If
Next
textReader.Close()
Return list
End Function
End Class

'Parent Class
Public Class Brand
Private m_brandName As String
<DisplayNameAttribute("Brand")>
Public Property BrandName As String
Get
Return m_brandName
End Get
Set(ByVal value As String)
m_brandName = value
End Set
End Property
Private m_vehicleTypes As IList(Of VehicleType)
Public Property VehicleTypes As IList(Of VehicleType)
Get
Return m_vehicleTypes
End Get
Set(ByVal value As IList(Of VehicleType))
m_vehicleTypes = value
End Set
End Property
Public Sub New(ByVal brandName As String)
m_brandName = brandName
End Sub
End Class

'Child Class
Public Class VehicleType
Private m_vehicleName As String
<DisplayNameAttribute("Vehicle Type")>
Public Property VehicleName As String

```

```

Get
Return m_vehicleName
End Get
Set(ByVal value As String)
m_vehicleName = value
End Set
End Property
Private m_models As IList(Of Model)
Public Property Models As IList(Of Model)
Get
Return m_models
End Get
Set(ByVal value As IList(Of Model))
m_models = value
End Set
End Property
Public Sub New(ByVal vehicle As String)
m_vehicleName = vehicle
End Sub
End Class
'Sub-child Class
Public Class Model
Private m_modelName As String
<DisplayNameAttribute("Model")>
Public Property ModelName As String
Get
Return m_modelName
End Get
Set(ByVal value As String)
m_modelName = value
End Set
End Property
Public Sub New(ByVal name As String)
m_modelName = name
End Sub
End Class
<XmlRootAttribute("BrandObjects")>
Public Class BrandObjects
<XmlElement("BrandObject")>
Public Property BrandObject As BrandObject()
End Class
Public Class BrandObject
Public Property BrandName As String
Public Property VehicleType As String
Public Property ModelName As String
End Class
End Namespace

```

UWP

```

using Syncfusion.XlsIO;
using System.Collections.Generic;
using System.ComponentModel;
using System.IO;
using System.Xml.Serialization;
namespace ImportFromNestedCollection

```



```

{
public sealed partial class MainPage : Page
{
public MainPage()
{
this.InitializeComponent();
}
//Button click to import data from nested collection to Excel worksheet.
private async void btnGenerateExcel_Click(object sender, RoutedEventArgs e)
{
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
IWorkbook workbook = excelEngine.Excel.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
IList<Brand> vehicles = GetVehicleDetails();
ExcelImportDataOptions importDataOptions = new ExcelImportDataOptions();
//Imports from 4th row.
importDataOptions.FirstRow = 4;
//Imports column headers.
importDataOptions.IncludeHeader = true;
//Set layout options.
importDataOptions.NestedDataLayoutOptions =
ExcelNestedDataLayoutOptions.Default;
//Import data from the nested collection.
worksheet.ImportData(vehicles, importDataOptions);
//Apply style to headers
worksheet["A1:C2"].Merge();
worksheet["A1"].Text = "Automobile Brands in the US";
worksheet.UsedRange.AutofitColumns();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ImportData";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
workbook.Close();
excelEngine.Dispose();
}
//Helper method to load data from XML file and add them in collections.
private static IList<Brand> GetVehicleDetails()
{
XmlSerializer deserializer = new XmlSerializer(typeof(BrandObjects));
//Read data from XML file.
Assembly assembly = typeof(MainPage).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("Sample.Data.ExportData.xml");
TextReader textReader = new StreamReader(fileStream);
BrandObjects brands = (BrandObjects)deserializer.Deserialize(textReader);
//Initialize parent collection to add data from XML file.
List<Brand> list = new List<Brand>();
string brandName = brands.BrandObject[0].BrandName;
string vehicleType = brands.BrandObject[0].VahicleType;

```

```

string modelName = brands.BrandObject[0].ModelName;
//Parent class
Brand brand = new Brand(brandName);
brand.VehicleTypes = new List<VehicleType>();
VehicleType vehicle = new VehicleType(vehicleType);
vehicle.Models = new List<Model>();
Model model = new Model(modelName);
brand.VehicleTypes.Add(vehicle);
list.Add(brand);
foreach (BrandObject brandObj in brands.BrandObject)
{
    if (brandName == brandObj.BrandName)
    {
        if (vehicleType == brandObj.VehicleType)
        {
            vehicle.Models.Add(new Model(brandObj.ModelName));
            continue;
        }
        else
        {
            vehicle = new VehicleType(brandObj.VehicleType);
            vehicle.Models = new List<Model>();
            vehicle.Models.Add(new Model(brandObj.ModelName));
            brand.VehicleTypes.Add(vehicle);
            vehicleType = brandObj.VehicleType;
        }
        continue;
    }
    else
    {
        brand = new Brand(brandObj.BrandName);
        vehicle = new VehicleType(brandObj.VehicleType);
        vehicle.Models = new List<Model>();
        vehicle.Models.Add(new Model(brandObj.ModelName));
        brand.VehicleTypes = new List<VehicleType>();
        brand.VehicleTypes.Add(vehicle);
        vehicleType = brandObj.VehicleType;
        list.Add(brand);
        brandName = brandObj.BrandName;
    }
}
textReader.Close();
return list;
}
}
//Parent Class
public class Brand
{
    private string m_brandName;
    [DisplayNameAttribute("Brand")]
    public string BrandName
    {
        get { return m_brandName; }
        set { m_brandName = value; }
    }
}
//Vehicle Types Collection
private IList<VehicleType> m_vehicleTypes;

```

```

public IList<VehicleType> VehicleTypes
{
    get { return m_vehicleTypes; }
    set { m_vehicleTypes = value; }
}
public Brand(string brandName)
{
    m_brandName = brandName;
}
}
//Child Class
public class VehicleType
{
    private string m_vehicleName;
    [DisplayNameAttribute("Vehicle Type")]
    public string VehicleName
    {
        get { return m_vehicleName; }
        set { m_vehicleName = value; }
    }
    //Models collection
    private IList<Model> m_models;
    public IList<Model> Models
    {
        get { return m_models; }
        set { m_models = value; }
    }
    public VehicleType(string vehicle)
    {
        m_vehicleName = vehicle;
    }
}
//Sub-child Class
public class Model
{
    private string m_modelName;
    [DisplayNameAttribute("Model")]
    public string ModelName
    {
        get { return m_modelName; }
        set { m_modelName = value; }
    }
    public Model(string name)
    {
        m_modelName = name;
    }
}
//Helper Classes
[XmlRootAttribute("BrandObjects")]
public class BrandObjects
{
    [XmlElement("BrandObject")]
    public BrandObject[] BrandObject { get; set; }
}
public class BrandObject
{
    public string BrandName { get; set; }
}

```

```

public string VehicleType { get; set; }
public string ModelName { get; set; }
}
}

```

ASP.NET CORE

```

using Syncfusion.XlsIO;
using System.Collections.Generic;
using System.ComponentModel;
using System.IO;
using System.Xml.Serialization;
namespace ImportFromNestedCollection
{
    class Program
    {
        static void Main(string[] args)
        {
            ImportData();
        }
        //Main method to import data from nested collection to Excel worksheet.
        private static void ImportData()
        {
            ExcelEngine excelEngine = new ExcelEngine();
            IApplication application = excelEngine.Excel;
            application.DefaultVersion = ExcelVersion.Excel2016;
            IWorkbook workbook = excelEngine.Excel.Workbooks.Create(1);
            IWorksheet worksheet = workbook.Worksheets[0];
            IList<Brand> vehicles = GetVehicleDetails();
            ExcelImportDataOptions importDataOptions = new ExcelImportDataOptions();
            //Imports from 4th row.
            importDataOptions.FirstRow = 4;
            //Imports column headers.
            importDataOptions.IncludeHeader = true;
            //Set layout options.
            importDataOptions.NestedDataLayoutOptions =
                ExcelNestedDataLayoutOptions.Default;
            //Import data from the nested collection.
            worksheet.ImportData(vehicles, importDataOptions);
            //Apply style to headers
            worksheet["A1:C2"].Merge();
            worksheet["A1"].Text = "Automobile Brands in the US";
            worksheet.UsedRange.AutofitColumns();
            //Saving the workbook as stream
            FileStream stream = new FileStream("ImportData.xlsx", FileMode.Create,
                FileAccess.ReadWrite);
            workbook.SaveAs(stream);
            stream.Dispose();
            workbook.Close();
            excelEngine.Dispose();
        }
        //Helper method to load data from XML file and add them in collections.
        private static IList<Brand> GetVehicleDetails()
        {
            XmlSerializer deserializer = new XmlSerializer(typeof(BrandObjects));
            //Read data from XML file.

```

```

FileStream stream = new FileStream(@"..\..\Data\ExportData.xml",
    FileMode.Open, FileAccess.Read);
TextReader textReader = new StreamReader(stream);
BrandObjects brands = (BrandObjects)deserializer.Deserialize(textReader);
//Initialize parent collection to add data from XML file.
List<Brand> list = new List<Brand>();
string brandName = brands.BrandObject[0].BrandName;
string vehicleType = brands.BrandObject[0].VahicleType;
string modelName = brands.BrandObject[0].ModelName;
//Parent class
Brand brand = new Brand(brandName);
brand.VehicleTypes = new List<VehicleType>();
VehicleType vehicle = new VehicleType(vehicleType);
vehicle.Models = new List<Model>();
Model model = new Model(modelName);
brand.VehicleTypes.Add(vehicle);
list.Add(brand);
foreach (BrandObject brandObj in brands.BrandObject)
{
    if (brandName == brandObj.BrandName)
    {
        if (vehicleType == brandObj.VahicleType)
        {
            vehicle.Models.Add(new Model(brandObj.ModelName));
            continue;
        }
        else
        {
            vehicle = new VehicleType(brandObj.VahicleType);
            vehicle.Models = new List<Model>();
            vehicle.Models.Add(new Model(brandObj.ModelName));
            brand.VehicleTypes.Add(vehicle);
            vehicleType = brandObj.VahicleType;
        }
        continue;
    }
    else
    {
        brand = new Brand(brandObj.BrandName);
        vehicle = new VehicleType(brandObj.VahicleType);
        vehicle.Models = new List<Model>();
        vehicle.Models.Add(new Model(brandObj.ModelName));
        brand.VehicleTypes = new List<VehicleType>();
        brand.VehicleTypes.Add(vehicle);
        vehicleType = brandObj.VahicleType;
        list.Add(brand);
        brandName = brandObj.BrandName;
    }
}
textReader.Close();
return list;
}
}
//Parent Class
public class Brand
{
    private string m_brandName;

```

```

[DisplayNameAttribute("Brand")]
public string BrandName
{
    get { return m_brandName; }
    set { m_brandName = value; }
}
//Vehicle Types Collection
private IList<VehicleType> m_vehicleTypes;
public IList<VehicleType> VehicleTypes
{
    get { return m_vehicleTypes; }
    set { m_vehicleTypes = value; }
}
public Brand(string brandName)
{
    m_brandName = brandName;
}
}
//Child Class
public class VehicleType
{
    private string m_vehicleName;
    [DisplayNameAttribute("Vehicle Type")]
    public string VehicleName
    {
        get { return m_vehicleName; }
        set { m_vehicleName = value; }
    }
    //Models collection
    private IList<Model> m_models;
    public IList<Model> Models
    {
        get { return m_models; }
        set { m_models = value; }
    }
    public VehicleType(string vehicle)
    {
        m_vehicleName = vehicle;
    }
}
//Sub-child Class
public class Model
{
    private string m_modelName;
    [DisplayNameAttribute("Model")]
    public string ModelName
    {
        get { return m_modelName; }
        set { m_modelName = value; }
    }
    public Model(string name)
    {
        m_modelName = name;
    }
}
//Helper Classes
[XmlRootAttribute("BrandObjects")]

```

```

public class BrandObjects
{
    [XmlElement("BrandObject")]
    public BrandObject[] BrandObject { get; set; }
}

public class BrandObject
{
    public string BrandName { get; set; }
    public string VehicleType { get; set; }
    public string ModelName { get; set; }
}
}

```

XAMARIN

```

using Syncfusion.XlsIO;
using System.Collections.Generic;
using System.ComponentModel;
using System.IO;
using System.Xml.Serialization;
namespace ImportFromNestedCollection
{
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            this.InitializeComponent();
        }
        //Button click to import data from nested collection to Excel worksheet.
        internal void OnButtonClicked(object sender, EventArgs e)
        {
            ExcelEngine excelEngine = new ExcelEngine();
            IApplication application = excelEngine.Excel;
            application.DefaultVersion = ExcelVersion.Excel2016;
            IWorkbook workbook = excelEngine.Excel.Workbooks.Create(1);
            IWorksheet worksheet = workbook.Worksheets[0];
            IList<Brand> vehicles = GetVehicleDetails();
            ExcelImportDataOptions importDataOptions = new ExcelImportDataOptions();
            //Imports from 4th row.
            importDataOptions.FirstRow = 4;
            //Imports column headers.
            importDataOptions.IncludeHeader = true;
            //Set layout options.
            importDataOptions.NestedDataLayoutOptions =
            ExcelNestedDataLayoutOptions.Default;
            //Import data from the nested collection.
            worksheet.ImportData(vehicles, importDataOptions);
            //Apply style to headers
            worksheet["A1:C2"].Merge();
            worksheet["A1"].Text = "Automobile Brands in the US";
            worksheet.UsedRange.AutofitColumns();
            //Save the document as file and view the saved document
            //The operation in SaveAndView under Xamarin varies between Windows Phone,
            Android, and iOS platforms. Refer to the xlsio/xamarin section for
            respective code samples

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Import
Data.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ImportData.xlsx",
"application/msexcel", stream);
}
workbook.Close();
excelEngine.Dispose();
}
//Helper method to load data from XML file and add them in collections.
private static IList<Brand> GetVehicleDetails()
{
XmlSerializer deserializer = new XmlSerializer(typeof(BrandObjects));
//Read data from XML file.
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("Sample.Data.ExportData.xml");
TextReader textReader = new StreamReader(fileStream);
BrandObjects brands = (BrandObjects)deserializer.Deserialize(textReader);
//Initialize parent collection to add data from XML file.
List<Brand> list = new List<Brand>();
string brandName = brands.BrandObject[0].BrandName;
string vehicleType = brands.BrandObject[0].VahicleType;
string modelName = brands.BrandObject[0].ModelName;
//Parent class
Brand brand = new Brand(brandName);
brand.VehicleTypes = new List<VehicleType>();
VehicleType vehicle = new VehicleType(vehicleType);
vehicle.Models = new List<Model>();
Model model = new Model(modelName);
brand.VehicleTypes.Add(vehicle);
list.Add(brand);
foreach (BrandObject brandObj in brands.BrandObject)
{
if (brandName == brandObj.BrandName)
{
if (vehicleType == brandObj.VahicleType)
{
vehicle.Models.Add(new Model(brandObj.ModelName));
continue;
}
else
{
vehicle = new VehicleType(brandObj.VahicleType);
vehicle.Models = new List<Model>();
vehicle.Models.Add(new Model(brandObj.ModelName));
brand.VehicleTypes.Add(vehicle);
vehicleType = brandObj.VahicleType;
}
continue;
}
else

```



```

{
    brand = new Brand(brandObj.BrandName);
    vehicle = new VehicleType(brandObj.VehicleType);
    vehicle.Models = new List<Model>();
    vehicle.Models.Add(new Model(brandObj.ModelName));
    brand.VehicleTypes = new List<VehicleType>();
    brand.VehicleTypes.Add(vehicle);
    vehicleType = brandObj.VehicleType;
    list.Add(brand);
    brandName = brandObj.BrandName;
}
}
textReader.Close();
return list;
}
}
//Parent Class
public class Brand
{
    private string m_brandName;
    [DisplayNameAttribute("Brand")]
    public string BrandName
    {
        get { return m_brandName; }
        set { m_brandName = value; }
    }
    //Vehicle Types Collection
    private IList<VehicleType> m_vehicleTypes;
    public IList<VehicleType> VehicleTypes
    {
        get { return m_vehicleTypes; }
        set { m_vehicleTypes = value; }
    }
    public Brand(string brandName)
    {
        m_brandName = brandName;
    }
}
//Child Class
public class VehicleType
{
    private string m_vehicleName;
    [DisplayNameAttribute("Vehicle Type")]
    public string VehicleName
    {
        get { return m_vehicleName; }
        set { m_vehicleName = value; }
    }
    //Models collection
    private IList<Model> m_models;
    public IList<Model> Models
    {
        get { return m_models; }
        set { m_models = value; }
    }
    public VehicleType(string vehicle)
    {

```

```

m_vehicleName = vehicle;
}
}
//Sub-child Class
public class Model
{
private string m_modelName;
[DisplayNameAttribute("Model")]
public string ModelName
{
get { return m_modelName; }
set { m_modelName = value; }
}
public Model(string name)
{
m_modelName = name;
}
}
//Helper Classes
[XmlRootAttribute("BrandObjects")]
public class BrandObjects
{
[XmlElement("BrandObject")]
public BrandObject[] BrandObject { get; set; }
}
public class BrandObject
{
public string BrandName { get; set; }
public string VehicleType { get; set; }
public string ModelName { get; set; }
}
}

```

The following screenshot represents the output document with Default layout option.

	A	B	C
1	Automobile Brands in the US		
2			
3			
4	Brand	Vehicle Type	Model
5	Ford	Cars	Fusion
6			Fiesta
7			Mustang
8		SUVs & Crossovers	Eco Sport
9			Escape
10			Flex
11		Trucks & Vans	Fusion
12			Transit Connect
13			F-150

Merge layout option

This option merges the cells in the column for each object while importing.

The following code snippet helps to import data with merged cells.

C#

```
importDataOptions.NestedDataLayoutOptions =
ExcelNestedDataLayoutOptions.Merge;
```

VB.NET

```
importDataOptions.NestedDataLayoutOptions =
ExcelNestedDataLayoutOptions.Merge
```

UWP

```
importDataOptions.NestedDataLayoutOptions =
ExcelNestedDataLayoutOptions.Merge;
```

ASP.NET CORE

```
importDataOptions.NestedDataLayoutOptions =
ExcelNestedDataLayoutOptions.Merge;
```

XAMARIN

```
importDataOptions.NestedDataLayoutOptions =
ExcelNestedDataLayoutOptions.Merge;
```

The following screenshot represents the output document with Merge layout option.

	A	B	C
1	Automobile Brands in the US		
2			
3			
4	Brand	Vehicle Type	Model
5	Ford	Cars	Fusion
6			Fiesta
7			Mustang
8		SUVs & Crossovers	Eco Sport
9			Escape
10			Flex
11		Trucks & Vans	Fusion
12			Transit Connect
13			F-150

[Repeat layout option](#)

This option repeats the parent records imported in all the rows.

The following code snippet helps to import data with repeated rows.

C#

```
importDataOptions.NestedDataLayoutOptions =
ExcelNestedDataLayoutOptions.Repeat;
```

VB.NET

```
importDataOptions.NestedDataLayoutOptions =
ExcelNestedDataLayoutOptions.Repeat
```

UWP

```
importDataOptions.NestedDataLayoutOptions =
ExcelNestedDataLayoutOptions.Repeat;
```

ASP.NET CORE

```
importDataOptions.NestedDataLayoutOptions =
ExcelNestedDataLayoutOptions.Repeat;
```

XAMARIN

```
importDataOptions.NestedDataLayoutOptions =
ExcelNestedDataLayoutOptions.Repeat;
```

The following screenshot represents the output document with Repeat layout option.

	A	B	C
1	Automobile Brands in the US		
2			
3			
4	Brand	Vehicle Type	Model
5	Ford	Cars	Fusion
6	Ford	Cars	Fiesta
7	Ford	Cars	Mustang
8	Ford	SUVs & Crossovers	Eco Sport
9	Ford	SUVs & Crossovers	Escape
10	Ford	SUVs & Crossovers	Flex
11	Ford	Trucks & Vans	Fusion
12	Ford	Trucks & Vans	Transit Connect
13	Ford	Trucks & Vans	F-150

Grouping Options**Import Data with Grouping option**

Hierarchical data imported into Excel worksheet must be shown its structure to analyze more flexible. In addition, if the data is grouped according to its level, it is easier to analyze. XlsIO supports to import hierarchical data from nested collection and group them while importing.

The following are the options that is supported to group on import.

- **Expand** – Imported data will be grouped and expanded.

- **Collapse** – Imported data will be grouped and collapsed at first level, by default.

In addition, `CollapseLevel` will group and collapse the mentioned level, upto the maximum of 8 levels.

The following code snippet illustrates how to import data directly from nested collection objects with collapse group option.

C#

```
using Syncfusion.XlsIO;
using System.Collections.Generic;
using System.ComponentModel;
using System.IO;
using System.Xml.Serialization;
namespace ImportFromNestedCollection
{
    class Program
    {
        static void Main(string[] args)
        {
            ImportData();
        }
        //Main method to import data from nested collection to Excel worksheet.
        private static void ImportData()
        {
            ExcelEngine excelEngine = new ExcelEngine();
            IApplication application = excelEngine.Excel;
            application.DefaultVersion = ExcelVersion.Excel2016;
            IWorkbook workbook = excelEngine.Excel.Workbooks.Create(1);
            IWorksheet worksheet = workbook.Worksheets[0];
            IList<Brand> vehicles = GetVehicleDetails();
            ExcelImportDataOptions importDataOptions = new ExcelImportDataOptions();
            //Imports from 4th row.
            importDataOptions.FirstRow = 4;
            //Imports column headers.
            importDataOptions.IncludeHeader = true;
            //Set layout options.
            importDataOptions.NestedDataLayoutOptions =
                ExcelNestedDataLayoutOptions.Default;
            //Set grouping option.
            importDataOptions.NestedDataGroupOptions =
                ExcelNestedDataGroupOptions.Collapse;
            //Set collapse level.
            //GroupingOption must set to 'Collapse' before applying 'CollapseLevel'.
            importDataOptions.CollapseLevel = 2;
            //Import data from the nested collection.
            worksheet.ImportData(vehicles, importDataOptions);
            //Apply style to headers
            worksheet["A1:C2"].Merge();
            worksheet["A1"].Text = "Automobile Brands in the US";
            worksheet.UsedRange.AutofitColumns();
            workbook.SaveAs("ImportData.xlsx");
            workbook.Close();
            excelEngine.Dispose();
        }
        //Helper method to load data from XML file and add them in collections.
        private static IList<Brand> GetVehicleDetails()
```

```

{
    XmlSerializer deserializer = new XmlSerializer(typeof(BrandObjects));
    //Read data from XML file.
    TextReader textReader = new StreamReader(@"..\..\Data\ExportData.xml");
    BrandObjects brands = (BrandObjects)deserializer.Deserialize(textReader);
    //Initialize parent collection to add data from XML file.
    List<Brand> list = new List<Brand>();
    string brandName = brands.BrandObject[0].BrandName;
    string vehicleType = brands.BrandObject[0].VahicleType;
    string modelName = brands.BrandObject[0].ModelName;
    //Parent class
    Brand brand = new Brand(brandName);
    brand.VehicleTypes = new List<VehicleType>();
    VehicleType vehicle = new VehicleType(vehicleType);
    vehicle.Models = new List<Model>();
    Model model = new Model(modelName);
    brand.VehicleTypes.Add(vehicle);
    list.Add(brand);
    foreach (BrandObject brandObj in brands.BrandObject)
    {
        if (brandName == brandObj.BrandName)
        {
            if (vehicleType == brandObj.VahicleType)
            {
                vehicle.Models.Add(new Model(brandObj.ModelName));
                continue;
            }
            else
            {
                vehicle = new VehicleType(brandObj.VahicleType);
                vehicle.Models = new List<Model>();
                vehicle.Models.Add(new Model(brandObj.ModelName));
                brand.VehicleTypes.Add(vehicle);
                vehicleType = brandObj.VahicleType;
            }
            continue;
        }
        else
        {
            brand = new Brand(brandObj.BrandName);
            vehicle = new VehicleType(brandObj.VahicleType);
            vehicle.Models = new List<Model>();
            vehicle.Models.Add(new Model(brandObj.ModelName));
            brand.VehicleTypes = new List<VehicleType>();
            brand.VehicleTypes.Add(vehicle);
            vehicleType = brandObj.VahicleType;
            list.Add(brand);
            brandName = brandObj.BrandName;
        }
    }
    textReader.Close();
    return list;
}
}
//Parent Class
public class Brand
{

```

```

private string m_brandName;
[DisplayNameAttribute("Brand")]
public string BrandName
{
    get { return m_brandName; }
    set { m_brandName = value; }
}
//Vehicle Types Collection
private IList<VehicleType> m_vehicleTypes;
public IList<VehicleType> VehicleTypes
{
    get { return m_vehicleTypes; }
    set { m_vehicleTypes = value; }
}
public Brand(string brandName)
{
    m_brandName = brandName;
}
}
//Child Class
public class VehicleType
{
    private string m_vehicleName;
    [DisplayNameAttribute("Vehicle Type")]
    public string VehicleName
    {
        get { return m_vehicleName; }
        set { m_vehicleName = value; }
    }
    //Models collection
    private IList<Model> m_models;
    public IList<Model> Models
    {
        get { return m_models; }
        set { m_models = value; }
    }
    public VehicleType(string vehicle)
    {
        m_vehicleName = vehicle;
    }
}
//Sub-child Class
public class Model
{
    private string m_modelName;
    [DisplayNameAttribute("Model")]
    public string ModelName
    {
        get { return m_modelName; }
        set { m_modelName = value; }
    }
    public Model(string name)
    {
        m_modelName = name;
    }
}
//Helper Classes

```

```
[XmlAttribute("BrandObjects")]
public class BrandObjects
{
    [XmlElement("BrandObject")]
    public BrandObject[] BrandObject { get; set; }
}

public class BrandObject
{
    public string BrandName { get; set; }
    public string VehicleType { get; set; }
    public string ModelName { get; set; }
}
}
```

VB.NET

```
Imports Syncfusion.XlsIO
Imports System.Collections.Generic
Imports System.ComponentModel
Imports System.IO
Imports System.Xml.Serialization
Namespace ImportFromNestedCollection
Class Program
Private Shared Sub Main(ByVal args As String())
    ImportData()
End Sub

'Main method to import data from nested collection to Excel worksheet.
Private Shared Sub ImportData()
    Dim excelEngine As ExcelEngine = New ExcelEngine()
    Dim application As IApplication = excelEngine.Excel
    application.DefaultVersion = ExcelVersion.Excel2016
    Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Create(1)
    Dim worksheet As IWorksheet = workbook.Worksheets(0)
    Dim vehicles As IList(Of Brand) = GetVehicleDetails()
    Dim importDataOptions As ExcelImportDataOptions = New
    ExcelImportDataOptions()
    'Imports from 4th row.
    importDataOptions.FirstRow = 4
    'Imports column headers.
    importDataOptions.IncludeHeader = True
    'Set layout options.
    importDataOptions.NestedDataLayoutOptions =
    ExcelNestedDataLayoutOptions.Default
    'Set grouping option.
    importDataOptions.NestedDataGroupOptions =
    ExcelNestedDataGroupOptions.Collapse
    'Set collapse level.
    'GroupingOption must set to 'Collapse' before applying 'CollapseLevel'.
    importDataOptions.CollapseLevel = 2;
    'Import data from the nested collection.
    worksheet.ImportData(vehicles, importDataOptions)
    'Apply style to headers
    worksheet("A1:C2").Merge()
    worksheet("A1").Text = "Automobile Brands in the US"
    worksheet.UsedRange.AutofitColumns()
    workbook.SaveAs("ImportData.xlsx")
End Sub
End Class
End Namespace
```



```

workbook.Close()
excelEngine.Dispose()
End Sub
'Helper method to load data from XML file and add them in collections.
Private Shared Function GetVehicleDetails() As IList(Of Brand)
Dim deserializer As XmlSerializer = New XmlSerializer(GetType(BrandObjects))
'Read data from XML file.
Dim textReader As TextReader = New StreamReader("../..\Data\ExportData.xml")
Dim brands As BrandObjects = CType(deserializer.Deserialize(textReader),
BrandObjects)
'Initialize parent collection to add data from XML file.
Dim list As List(Of Brand) = New List(Of Brand)()
Dim brandName As String = brands.BrandObject(0).BrandName
Dim vehicleType As String = brands.BrandObject(0).VahicleType
Dim modelName As String = brands.BrandObject(0).ModelName
'Parent class
Dim brand As Brand = New Brand(brandName)
brand.VehicleTypes = New List(Of VehicleType)()
Dim vehicle As VehicleType = New VehicleType(vehicleType)
vehicle.Models = New List(Of Model)()
Dim model As Model = New Model(modelName)
brand.VehicleTypes.Add(vehicle)
list.Add(brand)
For Each brandObj As BrandObject In brands.BrandObject
If brandName = brandObj.BrandName Then
If vehicleType = brandObj.VahicleType Then
vehicle.Models.Add(New Model(brandObj.ModelName))
Continue For
Else
vehicle = New VehicleType(brandObj.VahicleType)
vehicle.Models = New List(Of Model)()
vehicle.Models.Add(New Model(brandObj.ModelName))
brand.VehicleTypes.Add(vehicle)
vehicleType = brandObj.VahicleType
End If
Continue For
Else
brand = New Brand(brandObj.BrandName)
vehicle = New VehicleType(brandObj.VahicleType)
vehicle.Models = New List(Of Model)()
vehicle.Models.Add(New Model(brandObj.ModelName))
brand.VehicleTypes = New List(Of VehicleType)()
brand.VehicleTypes.Add(vehicle)
vehicleType = brandObj.VahicleType
list.Add(brand)
brandName = brandObj.BrandName
End If
Next
textReader.Close()
Return list
End Function
End Class
'Parent Class
Public Class Brand
Private m_brandName As String
<DisplayNameAttribute("Brand")>
Public Property BrandName As String

```

```

Get
Return m_brandName
End Get
Set(ByVal value As String)
m_brandName = value
End Set
End Property
Private m_vehicleTypes As IList(Of VehicleType)
Public Property VehicleTypes As IList(Of VehicleType)
Get
Return m_vehicleTypes
End Get
Set(ByVal value As IList(Of VehicleType))
m_vehicleTypes = value
End Set
End Property
Public Sub New(ByVal brandName As String)
m_brandName = brandName
End Sub
End Class
'Child Class
Public Class VehicleType
Private m_vehicleName As String
<DisplayNameAttribute("Vehicle Type")>
Public Property VehicleName As String
Get
Return m_vehicleName
End Get
Set(ByVal value As String)
m_vehicleName = value
End Set
End Property
Private m_models As IList(Of Model)
Public Property Models As IList(Of Model)
Get
Return m_models
End Get
Set(ByVal value As IList(Of Model))
m_models = value
End Set
End Property
Public Sub New(ByVal vehicle As String)
m_vehicleName = vehicle
End Sub
End Class
'Sub-child Class
Public Class Model
Private m_modelName As String
<DisplayNameAttribute("Model")>
Public Property ModelName As String
Get
Return m_modelName
End Get
Set(ByVal value As String)
m_modelName = value
End Set
End Property

```

```

Public Sub New(ByVal name As String)
    m_modelName = name
End Sub
End Class
<XmlRootAttribute("BrandObjects")>
Public Class BrandObjects
    <XmlElement("BrandObject")>
    Public Property BrandObject As BrandObject()
End Class
Public Class BrandObject
    Public Property BrandName As String
    Public Property VehicleType As String
    Public Property ModelName As String
End Class
End Namespace

```

UWP

```

using Syncfusion.XlsIO;
using System.Collections.Generic;
using System.ComponentModel;
using System.IO;
using System.Xml.Serialization;
namespace ImportFromNestedCollection
{
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }
        //Button click to import data from nested collection to Excel worksheet.
        private async void btnGenerateExcel_Click(object sender, RoutedEventArgs e)
        {
            ExcelEngine excelEngine = new ExcelEngine();
            IApplication application = excelEngine.Excel;
            application.DefaultVersion = ExcelVersion.Excel2016;
            IWorkbook workbook = excelEngine.Excel.Workbooks.Create(1);
            IWorksheet worksheet = workbook.Worksheets[0];
            IList<Brand> vehicles = GetVehicleDetails();
            ExcelImportDataOptions importDataOptions = new ExcelImportDataOptions();
            //Imports from 4th row.
            importDataOptions.FirstRow = 4;
            //Imports column headers.
            importDataOptions.IncludeHeader = true;
            //Set layout options.
            importDataOptions.NestedDataLayoutOptions =
                ExcelNestedDataLayoutOptions.Default;
            //Set grouping option.
            importDataOptions.NestedDataGroupOptions =
                ExcelNestedDataGroupOptions.Collapse;
            //Set collapse level.
            //GroupingOption must set to 'Collapse' before applying 'CollapseLevel'.
            importDataOptions.CollapseLevel = 2;
            //Import data from the nested collection.
            worksheet.ImportData(vehicles, importDataOptions);
        }
    }
}

```

```

//Apply style to headers
worksheet["A1:C2"].Merge();
worksheet["A1"].Text = "Automobile Brands in the US";
worksheet.UsedRange.AutofitColumns();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ImportData";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
workbook.Close();
excelEngine.Dispose();
}
//Helper method to load data from XML file and add them in collections.
private static IList<Brand> GetVehicleDetails()
{
XmlSerializer deserializer = new XmlSerializer(typeof(BrandObjects));
//Read data from XML file.
Assembly assembly = typeof(MainPage).GetTypeInfo().Assembly;
Stream fileStream =
assembly.GetManifestResourceStream("Sample.Data.ExportData.xml");
TextReader textReader = new StreamReader(fileStream);
BrandObjects brands = (BrandObjects)deserializer.Deserialize(textReader);
//Initialize parent collection to add data from XML file.
List<Brand> list = new List<Brand>();
string brandName = brands.BrandObject[0].BrandName;
string vehicleType = brands.BrandObject[0].VahicleType;
string modelName = brands.BrandObject[0].ModelName;
//Parent class
Brand brand = new Brand(brandName);
brand.VehicleTypes = new List<VehicleType>();
VehicleType vehicle = new VehicleType(vehicleType);
vehicle.Models = new List<Model>();
Model model = new Model(modelName);
brand.VehicleTypes.Add(vehicle);
list.Add(brand);
foreach (BrandObject brandObj in brands.BrandObject)
{
if (brandName == brandObj.BrandName)
{
if (vehicleType == brandObj.VahicleType)
{
vehicle.Models.Add(new Model(brandObj.ModelName));
continue;
}
else
{
vehicle = new VehicleType(brandObj.VahicleType);
vehicle.Models = new List<Model>();
vehicle.Models.Add(new Model(brandObj.ModelName));
brand.VehicleTypes.Add(vehicle);
vehicleType = brandObj.VahicleType;
}
}
}
}

```

```

        continue;
    }
    else
    {
        brand = new Brand(brandObj.BrandName);
        vehicle = new VehicleType(brandObj.VehicleType);
        vehicle.Models = new List<Model>();
        vehicle.Models.Add(new Model(brandObj.ModelName));
        brand.VehicleTypes = new List<VehicleType>();
        brand.VehicleTypes.Add(vehicle);
        vehicleType = brandObj.VehicleType;
        list.Add(brand);
        brandName = brandObj.BrandName;
    }
}
textReader.Close();
return list;
}
}

//Parent Class
public class Brand
{
    private string m_brandName;
    [DisplayNameAttribute("Brand")]
    public string BrandName
    {
        get { return m_brandName; }
        set { m_brandName = value; }
    }

    //Vehicle Types Collection
    private IList<VehicleType> m_vehicleTypes;
    public IList<VehicleType> VehicleTypes
    {
        get { return m_vehicleTypes; }
        set { m_vehicleTypes = value; }
    }

    public Brand(string brandName)
    {
        m_brandName = brandName;
    }
}

//Child Class
public class VehicleType
{
    private string m_vehicleName;
    [DisplayNameAttribute("Vehicle Type")]
    public string VehicleName
    {
        get { return m_vehicleName; }
        set { m_vehicleName = value; }
    }

    //Models collection
    private IList<Model> m_models;
    public IList<Model> Models
    {
        get { return m_models; }
        set { m_models = value; }
    }
}

```

```

}
public VehicleType(string vehicle)
{
    m_vehicleName = vehicle;
}
}
//Sub-child Class
public class Model
{
    private string m_modelName;
    [DisplayNameAttribute("Model")]
    public string ModelName
    {
        get { return m_modelName; }
        set { m_modelName = value; }
    }
    public Model(string name)
    {
        m_modelName = name;
    }
}
//Helper Classes
[XmlRootAttribute("BrandObjects")]
public class BrandObjects
{
    [XmlElement("BrandObject")]
    public BrandObject[] BrandObject { get; set; }
}
public class BrandObject
{
    public string BrandName { get; set; }
    public string VahicleType { get; set; }
    public string ModelName { get; set; }
}
}

```

ASP.NET CORE

```

using Syncfusion.XlsIO;
using System.Collections.Generic;
using System.ComponentModel;
using System.IO;
using System.Xml.Serialization;
namespace ImportFromNestedCollection
{
    class Program
    {
        static void Main(string[] args)
        {
            ImportData();
        }
        //Main method to import data from nested collection to Excel worksheet.
        private static void ImportData()
        {
            ExcelEngine excelEngine = new ExcelEngine();
            IApplication application = excelEngine.Excel;

```

```

application.DefaultVersion = ExcelVersion.Excel2016;
IWorkbook workbook = excelEngine.Excel.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
IList<Brand> vehicles = GetVehicleDetails();
ExcelImportDataOptions importDataOptions = new ExcelImportDataOptions();
//Imports from 4th row.
importDataOptions.FirstRow = 4;
//Imports column headers.
importDataOptions.IncludeHeader = true;
//Set layout options.
importDataOptions.NestedDataLayoutOptions =
ExcelNestedDataLayoutOptions.Default;
//Set grouping option.
importDataOptions.NestedDataGroupOptions =
ExcelNestedDataGroupOptions.Collapse;
//Set collapse level.
//GroupingOption must set to 'Collapse' before applying 'CollapseLevel'.
importDataOptions.CollapseLevel = 2;
//Import data from the nested collection.
worksheet.ImportData(vehicles, importDataOptions);
//Apply style to headers
worksheet["A1:C2"].Merge();
worksheet["A1"].Text = "Automobile Brands in the US";
worksheet.UsedRange.AutofitColumns();
//Saving the workbook as stream
FileStream stream = new FileStream("ImportData.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
workbook.Close();
excelEngine.Dispose();
}
//Helper method to load data from XML file and add them in collections.
private static IList<Brand> GetVehicleDetails()
{
XmlSerializer deserializer = new XmlSerializer(typeof(BrandObjects));
//Read data from XML file.
FileStream stream = new FileStream(@"..\..\Data\ExportData.xml",
FileMode.Open, FileAccess.Read);
TextReader textReader = new StreamReader(stream);
BrandObjects brands = (BrandObjects)deserializer.Deserialize(textReader);
//Initialize parent collection to add data from XML file.
List<Brand> list = new List<Brand>();
string brandName = brands.BrandObject[0].BrandName;
string vehicleType = brands.BrandObject[0].VahicleType;
string modelName = brands.BrandObject[0].ModelName;
//Parent class
Brand brand = new Brand(brandName);
brand.VehicleTypes = new List<VehicleType>();
VehicleType vehicle = new VehicleType(vehicleType);
vehicle.Models = new List<Model>();
Model model = new Model(modelName);
brand.VehicleTypes.Add(vehicle);
list.Add(brand);
foreach (BrandObject brandObj in brands.BrandObject)
{
if (brandName == brandObj.BrandName)

```

```

{
    if (vehicleType == brandObj.VehicleType)
    {
        vehicle.Models.Add(new Model(brandObj.ModelName));
        continue;
    }
    else
    {
        vehicle = new VehicleType(brandObj.VehicleType);
        vehicle.Models = new List<Model>();
        vehicle.Models.Add(new Model(brandObj.ModelName));
        brand.VehicleTypes.Add(vehicle);
        vehicleType = brandObj.VehicleType;
    }
    continue;
}
else
{
    brand = new Brand(brandObj.BrandName);
    vehicle = new VehicleType(brandObj.VehicleType);
    vehicle.Models = new List<Model>();
    vehicle.Models.Add(new Model(brandObj.ModelName));
    brand.VehicleTypes = new List<VehicleType>();
    brand.VehicleTypes.Add(vehicle);
    vehicleType = brandObj.VehicleType;
    list.Add(brand);
    brandName = brandObj.BrandName;
}
}
textReader.Close();
return list;
}
}
//Parent Class
public class Brand
{
    private string m_brandName;
    [DisplayNameAttribute("Brand")]
    public string BrandName
    {
        get { return m_brandName; }
        set { m_brandName = value; }
    }
    //Vehicle Types Collection
    private IList<VehicleType> m_vehicleTypes;
    public IList<VehicleType> VehicleTypes
    {
        get { return m_vehicleTypes; }
        set { m_vehicleTypes = value; }
    }
    public Brand(string brandName)
    {
        m_brandName = brandName;
    }
}
//Child Class
public class VehicleType

```



```

{
    private string m_vehicleName;
    [DisplayNameAttribute("Vehicle Type")]
    public string VehicleName
    {
        get { return m_vehicleName; }
        set { m_vehicleName = value; }
    }
    //Models collection
    private IList<Model> m_models;
    public IList<Model> Models
    {
        get { return m_models; }
        set { m_models = value; }
    }
    public VehicleType(string vehicle)
    {
        m_vehicleName = vehicle;
    }
}
//Sub-child Class
public class Model
{
    private string m_modelName;
    [DisplayNameAttribute("Model")]
    public string ModelName
    {
        get { return m_modelName; }
        set { m_modelName = value; }
    }
    public Model(string name)
    {
        m_modelName = name;
    }
}
//Helper Classes
[XmlRootAttribute("BrandObjects")]
public class BrandObjects
{
    [XmlElement("BrandObject")]
    public BrandObject[] BrandObject { get; set; }
}
public class BrandObject
{
    public string BrandName { get; set; }
    public string VehicleType { get; set; }
    public string ModelName { get; set; }
}
}

```

XAMARIN

```

using Syncfusion.XlsIO;
using System.Collections.Generic;
using System.ComponentModel;
using System.IO;

```

```

using System.Xml.Serialization;
namespace ImportFromNestedCollection
{
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            this.InitializeComponent();
        }
        //Button click to import data from nested collection to Excel worksheet.
        internal void OnButtonClicked(object sender, EventArgs e)
        {
            ExcelEngine excelEngine = new ExcelEngine();
            IApplication application = excelEngine.Excel;
            application.DefaultVersion = ExcelVersion.Excel2016;
            IWorkbook workbook = excelEngine.Excel.Workbooks.Create(1);
            IWorksheet worksheet = workbook.Worksheets[0];
            IList<Brand> vehicles = GetVehicleDetails();
            ExcelImportDataOptions importDataOptions = new ExcelImportDataOptions();
            //Imports from 4th row.
            importDataOptions.FirstRow = 4;
            //Imports column headers.
            importDataOptions.IncludeHeader = true;
            //Set layout options.
            importDataOptions.NestedDataLayoutOptions =
                ExcelNestedDataLayoutOptions.Default;
            //Set grouping option.
            importDataOptions.NestedDataGroupOptions =
                ExcelNestedDataGroupOptions.Collapse;
            //Set collapse level.
            //GroupingOption must set to 'Collapse' before applying 'CollapseLevel'.
            importDataOptions.CollapseLevel = 2;
            //Import data from the nested collection.
            worksheet.ImportData(vehicles, importDataOptions);
            //Apply style to headers
            worksheet["A1:C2"].Merge();
            worksheet["A1"].Text = "Automobile Brands in the US";
            worksheet.UsedRange.AutofitColumns();
            //Save the document as file and view the saved document
            //The operation in SaveAndView under Xamarin varies between Windows Phone,
            //Android, and iOS platforms. Refer to the xlsio/xamarin section for
            //respective code samples
            if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
                TargetPlatform.Windows)
            {
                Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Import
                Data.xlsx", "application/msexcel", stream);
            }
            else
            {
                Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ImportData.xlsx",
                "application/msexcel", stream);
            }
            workbook.Close();
            excelEngine.Dispose();
        }
        //Helper method to load data from XML file and add them in collections.

```

```

private static IList<Brand> GetVehicleDetails()
{
    XmlSerializer deserializer = new XmlSerializer(typeof(BrandObjects));
    //Read data from XML file.
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream fileStream =
        assembly.GetManifestResourceStream("Sample.Data.ExportData.xml");
    TextReader textReader = new StreamReader(fileStream);
    BrandObjects brands = (BrandObjects)deserializer.Deserialize(textReader);
    //Initialize parent collection to add data from XML file.
    List<Brand> list = new List<Brand>();
    string brandName = brands.BrandObject[0].BrandName;
    string vehicleType = brands.BrandObject[0].VahicleType;
    string modelName = brands.BrandObject[0].ModelName;
    //Parent class
    Brand brand = new Brand(brandName);
    brand.VehicleTypes = new List<VehicleType>();
    VehicleType vehicle = new VehicleType(vehicleType);
    vehicle.Models = new List<Model>();
    Model model = new Model(modelName);
    brand.VehicleTypes.Add(vehicle);
    list.Add(brand);
    foreach (BrandObject brandObj in brands.BrandObject)
    {
        if (brandName == brandObj.BrandName)
        {
            if (vehicleType == brandObj.VahicleType)
            {
                vehicle.Models.Add(new Model(brandObj.ModelName));
                continue;
            }
            else
            {
                vehicle = new VehicleType(brandObj.VahicleType);
                vehicle.Models = new List<Model>();
                vehicle.Models.Add(new Model(brandObj.ModelName));
                brand.VehicleTypes.Add(vehicle);
                vehicleType = brandObj.VahicleType;
            }
            continue;
        }
        else
        {
            brand = new Brand(brandObj.BrandName);
            vehicle = new VehicleType(brandObj.VahicleType);
            vehicle.Models = new List<Model>();
            vehicle.Models.Add(new Model(brandObj.ModelName));
            brand.VehicleTypes = new List<VehicleType>();
            brand.VehicleTypes.Add(vehicle);
            vehicleType = brandObj.VahicleType;
            list.Add(brand);
            brandName = brandObj.BrandName;
        }
    }
    textReader.Close();
    return list;
}

```

```

}
//Parent Class
public class Brand
{
private string m_brandName;
[DisplayNameAttribute("Brand")]
public string BrandName
{
get { return m_brandName; }
set { m_brandName = value; }
}
//Vehicle Types Collection
private IList<VehicleType> m_vehicleTypes;
public IList<VehicleType> VehicleTypes
{
get { return m_vehicleTypes; }
set { m_vehicleTypes = value; }
}
public Brand(string brandName)
{
m_brandName = brandName;
}
}
//Child Class
public class VehicleType
{
private string m_vehicleName;
[DisplayNameAttribute("Vehicle Type")]
public string VehicleName
{
get { return m_vehicleName; }
set { m_vehicleName = value; }
}
//Models collection
private IList<Model> m_models;
public IList<Model> Models
{
get { return m_models; }
set { m_models = value; }
}
public VehicleType(string vehicle)
{
m_vehicleName = vehicle;
}
}
//Sub-child Class
public class Model
{
private string m_modelName;
[DisplayNameAttribute("Model")]
public string ModelName
{
get { return m_modelName; }
set { m_modelName = value; }
}
public Model(string name)
{

```

```

m_modelName = name;
}
}
//Helper Classes
[XmlAttribute("BrandObjects")]
public class BrandObjects
{
    [XmlElement("BrandObject")]
    public BrandObject[] BrandObject { get; set; }
}
public class BrandObject
{
    public string BrandName { get; set; }
    public string VahicleType { get; set; }
    public string ModelName { get; set; }
}
}

```

The following screenshot represents the output document of Grouped data imported from nested collection and collapsed at level 2.

1	2	3	A	B	C
	1		Automobile Brands in the US		
	2				
	3				
	4		Brand	Vehicle Type	Model
	5		Ford	Cars	Fusion
	8			SUVs & Crossovers	Eco Sport
	11			Trucks & Vans	Fusion
	14			Electrics	Fusion Hybrid SE
	17			Commercial Vehicles	Stripped Chassis
	20			Performance Vehicles	F-150 Raptor
	23		Chevrolet	Crossover and SUVs	Trax
	29			Cars	Spark
	34			Electrics	Volt
	36			Performance Vehicles	Camaro
	42			Trucks & Vans	Colorado
	45			Commercial Vehicles	Colorado

[Import Data from Collection Objects with hyperlink](#)

Essential XlsIO allows you to import images, data with URLs, and data with mail IDs as hyperlinks from various data sources binded in Collection Objects as shown below

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
}

```

```
//Import the data to worksheet
IList<Company> reports = GetCompanyDetails();
worksheet.ImportData(reports, 2, 1, false);
workbook.SaveAs("ImportFromBO.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Import the data to worksheet
Dim reports As IList(Of Company) = GetCompanyDetails()
worksheet.ImportData(reports, 2, 1, False)
workbook.SaveAs("ImportFromBO.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Import the data to worksheet
IList<Company> reports = GetCompanyDetails();
worksheet.ImportData(reports, 2, 1, false);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ImportFromBO";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Import the data to worksheet
IList<Company> reports = GetCompanyDetails();
worksheet.ImportData(reports, 2, 1, false);
//Saving the workbook as stream
```

```

FileStream stream = new FileStream("ImportFromBO.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Import the data to worksheet
    IList<Company> reports = GetCompanyDetails();
    worksheet.ImportData(reports, 2, 1, false);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Import
        FromBO.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ImportFromBO.xlsx"
        , "application/msexcel", stream);
    }
}

```

The following code snippet provides supporting methods and classes for the previous code.

C#

```

//Gets a list of company details
private List<Company> GetCompanyDetails()
{
    List<Company> companyList = new List<Company>();
    Company company = new Company();
    company.Name = "Syncfusion";
    Hyperlink link = new Hyperlink("https://www.syncfusion.com", "", "",
    "Syncfusion", ExcelHyperLinkType.Url, null);
    company.Link = link;
    companyList.Add(company);
    company = new Company();
    company.Name = "Microsoft";
    link = new Hyperlink("https://www.microsoft.com", "", "", "Microsoft",
    ExcelHyperLinkType.Url, null);
}

```

```

company.Link = link;
companyList.Add(company);
company = new Company();
company.Name = "Google";
link = new Hyperlink("https://www.google.com", "", "", "Google",
ExcelHyperLinkType.Url, null);
company.Link = link;
companyList.Add(company);
return companyList;
}

public class Hyperlink : IHyperLink
{
public IApplication Application { get; }
public object Parent { get; }
public string Address { get; set; }
public string Name { get; }
public IRange Range { get; }
public string ScreenTip { get; set; }
public string SubAddress { get; set; }
public string TextToDisplay { get; set; }
public ExcelHyperLinkType Type { get; set; }
public IShape Shape { get; }
public ExcelHyperlinkAttachedType AttachedType { get; }
public byte[] Image { get; set; }
public Hyperlink(string address, string subAddress, string screenTip, string
textToDisplay, ExcelHyperLinkType type, byte[] image)
{
Address = address;
ScreenTip = screenTip;
SubAddress = subAddress;
TextToDisplay = textToDisplay;
Type = type;
Image = image;
}
}

public class Company
{
public string Name { get; set; }
public Hyperlink Link { get; set; }
}

```

VB.NET

```

'Gets a list of company details
Private Function GetCompanyDetails() As List(Of Company)
Dim companyList As List(Of Company) = New List(Of Company) ()
Dim company As Company = New Company()
company.Name = "Syncfusion"
Dim link As Hyperlink = New Hyperlink("https://www.syncfusion.com", "", "",
"Syncfusion", ExcelHyperLinkType.Url, Nothing)
company.Link = link
companyList.Add(company)
company = New Company()
company.Name = "Microsoft"
link = New Hyperlink("https://www.microsoft.com", "", "", "Microsoft",
ExcelHyperLinkType.Url, Nothing)

```



```

company.Link = link
companyList.Add(company)
company = New Company()
company.Name = "Google"
link = New Hyperlink("https://www.google.com", "", "", "Google",
ExcelHyperLinkType.Url, Nothing)
company.Link = link
companyList.Add(company)
Return companyList
End Function
Public Class Hyperlink
Inherits IHyperLink
Public ReadOnly Property Application As IApplication
Public ReadOnly Property Parent As Object
Public Property Address As String
Public ReadOnly Property Name As String
Public ReadOnly Property Range As IRange
Public Property ScreenTip As String
Public Property SubAddress As String
Public Property TextToDisplay As String
Public Property Type As ExcelHyperLinkType
Public ReadOnly Property Shape As IShape
Public ReadOnly Property AttachedType As ExcelHyperlinkAttachedType
Public Property Image As Byte()
Public Sub New(ByVal address As String, ByVal subAddress As String, ByVal
screenTip As String, ByVal textToDisplay As String, ByVal type As
ExcelHyperLinkType, ByVal image As Byte())
Address = address
ScreenTip = screenTip
SubAddress = subAddress
TextToDisplay = textToDisplay
Type = type
Image = image
End Sub
End Class
Public Class Company
Public Property Name As String
Public Property Link As Hyperlink
End Class

```

UWP

```

//Gets a list of company details
private List<Company> GetCompanyDetails()
{
List<Company> companyList = new List<Company>();
Company company = new Company();
company.Name = "Syncfusion";
Hyperlink link = new Hyperlink("https://www.syncfusion.com", "", "",
"Syncfusion", ExcelHyperLinkType.Url, null);
company.Link = link;
companyList.Add(company);
company = new Company();
company.Name = "Microsoft";
link = new Hyperlink("https://www.microsoft.com", "", "", "Microsoft",
ExcelHyperLinkType.Url, null);

```

```

company.Link = link;
companyList.Add(company);
company = new Company();
company.Name = "Google";
link = new Hyperlink("https://www.google.com", "", "", "Google",
ExcelHyperLinkType.Url, null);
company.Link = link;
companyList.Add(company);
return companyList;
}

public class Hyperlink : IHyperLink
{
public IApplication Application { get; }
public object Parent { get; }
public string Address { get; set; }
public string Name { get; }
public IRange Range { get; }
public string ScreenTip { get; set; }
public string SubAddress { get; set; }
public string TextToDisplay { get; set; }
public ExcelHyperLinkType Type { get; set; }
public IShape Shape { get; }
public ExcelHyperlinkAttachedType AttachedType { get; }
public byte[] Image { get; set; }
public Hyperlink(string address, string subAddress, string screenTip, string
textToDisplay, ExcelHyperLinkType type, byte[] image)
{
Address = address;
ScreenTip = screenTip;
SubAddress = subAddress;
TextToDisplay = textToDisplay;
Type = type;
Image = image;
}
}

public class Company
{
public string Name { get; set; }
public Hyperlink Link { get; set; }
}

```

ASP.NET CORE

```

//Gets a list of company details
private List<Company> GetCompanyDetails()
{
List<Company> companyList = new List<Company>();
Company company = new Company();
company.Name = "Syncfusion";
Hyperlink link = new Hyperlink("https://www.syncfusion.com", "", "",
"Syncfusion", ExcelHyperLinkType.Url, null);
company.Link = link;
companyList.Add(company);
company = new Company();
company.Name = "Microsoft";
}

```

```

link = new Hyperlink("https://www.microsoft.com", "", "", "Microsoft",
ExcelHyperLinkType.Url, null);
company.Link = link;
companyList.Add(company);
company = new Company();
company.Name = "Google";
link = new Hyperlink("https://www.google.com", "", "", "Google",
ExcelHyperLinkType.Url, null);
company.Link = link;
companyList.Add(company);
return companyList;
}
public class Hyperlink : IHyperLink
{
public IApplication Application { get; }
public object Parent { get; }
public string Address { get; set; }
public string Name { get; }
public IRange Range { get; }
public string ScreenTip { get; set; }
public string SubAddress { get; set; }
public string TextToDisplay { get; set; }
public ExcelHyperLinkType Type { get; set; }
public IShape Shape { get; }
public ExcelHyperlinkAttachedType AttachedType { get; }
public byte[] Image { get; set; }
public Hyperlink(string address, string subAddress, string screenTip, string
textToDisplay, ExcelHyperLinkType type, byte[] image)
{
Address = address;
ScreenTip = screenTip;
SubAddress = subAddress;
TextToDisplay = textToDisplay;
Type = type;
Image = image;
}
}
public class Company
{
public string Name { get; set; }
public Hyperlink Link { get; set; }
}

```

XAMARIN

```

//Gets a list of company details
private List<Company> GetCompanyDetails()
{
List<Company> companyList = new List<Company>();
Company company = new Company();
company.Name = "Syncfusion";
Hyperlink link = new Hyperlink("https://www.syncfusion.com", "", "",
"Syncfusion", ExcelHyperLinkType.Url, null);
company.Link = link;
companyList.Add(company);
company = new Company();
}

```

```

company.Name = "Microsoft";
link = new Hyperlink("https://www.microsoft.com", "", "", "Microsoft",
ExcelHyperLinkType.Url, null);
company.Link = link;
companyList.Add(company);
company = new Company();
company.Name = "Google";
link = new Hyperlink("https://www.google.com", "", "", "Google",
ExcelHyperLinkType.Url, null);
company.Link = link;
companyList.Add(company);
return companyList;
}

public class Hyperlink : IHyperLink
{
    public IApplication Application { get; }
    public object Parent { get; }
    public string Address { get; set; }
    public string Name { get; }
    public IRange Range { get; }
    public string ScreenTip { get; set; }
    public string SubAddress { get; set; }
    public string TextToDisplay { get; set; }
    public ExcelHyperLinkType Type { get; set; }
    public IShape Shape { get; }
    public ExcelHyperlinkAttachedType AttachedType { get; }
    public byte[] Image { get; set; }
    public Hyperlink(string address, string subAddress, string screenTip, string
textToDisplay, ExcelHyperLinkType type, byte[] image)
    {
        Address = address;
        ScreenTip = screenTip;
        SubAddress = subAddress;
        TextToDisplay = textToDisplay;
        Type = type;
        Image = image;
    }
}

public class Company
{
    public string Name { get; set; }
    public Hyperlink Link { get; set; }
}

```

Import Data from Array

The following code snippet shows how to import array of data into a worksheet using **ImportArray** method.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
}

```

```
//Initialize the Object Array
object[] array = new object[4] { "Total Income", "Actual Expense", "Expected
Expenses", "Profit" };
//Import the Object Array to Sheet
worksheet.ImportArray(array, 1, 1, false);
workbook.SaveAs("ImportFromDT.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Initialize the Array Object
Dim array() As Object = New Object() {"Total Income", "Actual Expense",
"Expected Expenses", "Profit"}
'Import the Array Object to Sheet
worksheet.ImportArray(array, 1, 1, False)
workbook.SaveAs("ImportFromDT.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Initialize the Object Array
object[] array = new object[4] { "Total Income", "Actual Expense", "Expected
Expenses", "Profit" };
//Import the Object Array to Sheet
worksheet.ImportArray(array, 1, 1, false);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ImportFromDT";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
```

```

IWorksheet worksheet = workbook.Worksheets[0];
//Initialize the Object Array
object[] array = new object[4] { "Total Income", "Actual Expense", "Expected Expenses", "Profit" };
//Import the Object Array to Sheet
worksheet.ImportArray(array, 1, 1, false);
//Saving the workbook as stream
FileStream stream = new FileStream("ImportFromDT.xlsx", FileMode.Create, FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Initialize the Object Array
    object[] array = new object[4] { "Total Income", "Actual Expense", "Expected Expenses", "Profit" };
    //Import the Object Array to Sheet
    worksheet.ImportArray(array, 1, 1, false);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone, Android, and iOS platforms. Refer to the xlsio/xamarin section for respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("ImportFromDT.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ImportFromDT.xlsx", "application/msexcel", stream);
    }
}

```

Exporting from Worksheet to Data Table

XlsIO allows to export the sheet data to a **DataTable** by using the **ExportDataTable()** method. This method provides various options that allows to export data with specific requirement through **ExcelExportDataTableOptions**.

The following code snippet illustrates on how to export data from worksheet to Data grid using **DataTable**.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Export3.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Read data from the worksheet and Export to the DataTable
    DataTable customersTable = worksheet.ExportDataTable(worksheet.UsedRange,
        ExcelExportDataTableOptions.ColumnNames);
    //Binding exported DataTable to data grid, likewise it can binded to any
    //user interface control which supports binding
    DataGrid dataGrid = new DataGrid();
    dataGrid.DataSource = customersTable;
    workbook.SaveAs("ExportToGrid.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Read data from the worksheet and Export to the DataTable
Dim customersTable As DataTable = sheet.ExportDataTable(sheet.UsedRange,
    ExcelExportDataTableOptions.ColumnNames)
'Binding exported DataTable to data grid, likewise it can binded to any
'user interface control which supports binding
Dim dataGrid As DataGrid = New DataGrid
dataGrid.DataSource = customersTable
workbook.SaveAs("ExportToGrid.xlsx")
End Using

```

UWP

```

//XlsIO supports exporting of data from worksheet to data table in Windows
Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Core (2.0 onwards) platforms
alone.

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Read data from the worksheet and Export to the DataTable
    DataTable customersTable = worksheet.ExportDataTable(worksheet.UsedRange,
        ExcelExportDataTableOptions.ColumnNames);
    //Saving the workbook as stream
}

```

```

FileStream stream = new FileStream("ExportToDT.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}
//XlsIO supports binding of exported data table to data grid in Windows
Forms, WPF, ASP.NET and ASP.NET MVC platforms alone.

```

XAMARIN

```

//XlsIO supports exporting of data from worksheet to data table in Windows
Forms, WPF, ASP.NET, ASP.NET MVC and ASP.NET Core (2.0 onwards) platforms
alone.

```

Exporting from Worksheet to Collection Objects

XlsIO allows to export the sheet data to a **Collection Objects** by using the **ExportData<T>()** method.

The following code snippet illustrates on how to export worksheet data into Collection Objects using **ExportData<T>**.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Export worksheet data into Collection Objects
    List<Report> collectionObjects = worksheet.ExportData<Report>(1, 1, 10, 3);
    workbook.SaveAs("CollectionObjects.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Export worksheet data into Collection Objects
Dim collectionObjects As List(Of Report) = worksheet.ExportData(Of
Report)(1, 1, 10, 3)
workbook.SaveAs("CollectionObjects.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
}

```



```

openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Export worksheet data into Collection Objects
List<Report> collectionObjects = worksheet.ExportData<Report>(1, 1, 10, 3);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "CollectionObjects";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Export worksheet data into Collection Objects
    List<Report> collectionObjects = worksheet.ExportData<Report>(1, 1, 10, 3);
    //Saving the workbook as stream
    FileStream stream = new FileStream("CollectionObjects.xlsx",
    FileMode.Create, FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
    ple.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Export worksheet data into Collection Objects
    List<Report> collectionObjects = worksheet.ExportData<Report>(1, 1, 10, 3);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();

```

```

workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android, and iOS platforms. Refer to the xlsio/xamarin section for
//respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("CollectionObjects.xlsx", "application/msexcel", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("CollectionObjects.xlsx", "application/msexcel", stream);
}
}

```

The following code snippet provides supporting class for the above code. Here, the attributes `DisplayNameAttribute` and `Bindable` are used.

- [DisplayNameAttribute](#) - to match the column headers with set of properties while exporting.
- [BindableAttribute](#) - to skip a property while exporting.

C#

```

public class Report
{
    [DisplayNameAttribute("Sales Person Name")]
    public string SalesPerson { get; set; }
    [Bindable(false)]
    public string SalesJanJun { get; set; }
    public string SalesJulDec { get; set; }
    public Report()
    {
    }
}

```

VB.NET

```

Public Class Report
    Private m_SalesPerson As String
    Private m_SalesJanJun As String
    Private m_SalesJulDec As String
    <DisplayNameAttribute("Sales Person Name")>
    Public Property SalesPerson() As String
    Get
    Return m_SalesPerson
    End Get
    Set(value As String)
    m_SalesPerson = Value
    End Set
    End Property

```

```

<Bindable(False)>
Public Property SalesJanJun() As String
Get
Return m_SalesJanJun
End Get
Set(value As String)
m_SalesJanJun = Value
End Set
End Property
Public Property SalesJulDec() As String
Get
Return m_SalesJulDec
End Get
Set(value As String)
m_SalesJulDec = Value
End Set
End Property
End Class

```

UWP

```

public class Report
{
[DisplayNameAttribute("Sales Person Name")]
public string SalesPerson { get; set; }
[Bindable(false)]
public string SalesJanJun { get; set; }
public string SalesJulDec { get; set; }
public Report()
{
}
}

```

ASP.NET CORE

```

public class Report
{
[DisplayNameAttribute("Sales Person Name")]
public string SalesPerson { get; set; }
[Bindable(false)]
public string SalesJanJun { get; set; }
public string SalesJulDec { get; set; }
public Report()
{
}
}

```

XAMARIN

```

public class Report
{
[DisplayNameAttribute("Sales Person Name")]
public string SalesPerson { get; set; }
[Bindable(false)]
public string SalesJanJun { get; set; }

```

```
public string SalesJulDec { get; set; }
public Report()
{
}
}
```

Export data from Excel to Nested Class Objects

XlsIO allows to export worksheet data to nested class objects. A new overload to the existing `ExportData<T>()` method helps to achieve this requirement by mapping column headers with class properties.

Let's consider the input Excel document has the data as shown in the below screenshot.

	A	B	C	D	E
1	Customer ID	Customer Name	Customer Age	Order ID	Order Price
2	1324	Ashley Feathers	39	Z2340	\$ 450.00
3	1453	Abraham Wood	45	S2356	\$ 960.00
4	1239	Morris Jackson	35	A2345	\$ 320.00
5	1238	Robert Clay	40	B2345	\$ 550.00
6	1543	Victoria Sanders	45	A3423	\$ 600.00

The following code illustrates how to export data from Excel worksheet to nested class objects with column headers mapping collection.

C#

```
using Syncfusion.XlsIO;
using System.Collections.Generic;
namespace ImportFromNestedCollection
{
    class Program
    {
        static void Main(string[] args)
        {
            ExportData();
        }
        //Main method to Export data from worksheet to nested class objects.
        private static void ExportData()
        {
            using (ExcelEngine excelEngine = new ExcelEngine())
            {
                IApplication application = excelEngine.Excel;
                application.DefaultVersion = ExcelVersion.Excel2013;
                IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
                IWorksheet worksheet = workbook.Worksheets[0];
                //Map column headers in worksheet with class properties.
                Dictionary<string, string> mappingProperties = new Dictionary<string, string>();
                mappingProperties.Add("Customer ID", "CustId");
                mappingProperties.Add("Customer Name", "CustName");
                mappingProperties.Add("Customer Age", "CustAge");
                mappingProperties.Add("Order ID", "CustOrder.Order_Id");
                mappingProperties.Add("Order Price", "CustOrder.Price");
                //Export worksheet data into nested class Objects.
            }
        }
    }
}
```

```

List<Customer> nestedClassObjects = worksheet.ExportData<Customer>(1, 1, 10,
5, mappingProperties);
workbook.SaveAs("NestedClassObjects.xlsx");
}
}
}
//Customer details class
public partial class Customer
{
    public int CustId { get; set; }
    public string CustName { get; set; }
    public int CustAge { get; set; }
    public Order CustOrder { get; set; }
    public Customer()
    {
    }
}
//Order details class
public partial class Order
{
    public int Order_Id { get; set; }
    public double Price { get; set; }
    public Order()
    {
    }
}
}

```

VB.NET

```

Imports Syncfusion.XlsIO
Imports System.Collections.Generic
Namespace ImportFromNestedCollection
Class Program
    Private Shared Sub Main(ByVal args As String())
        ExportData()
    End Sub
    'Main method to Export data from worksheet to nested class objects.
    Private Shared Sub ExportData()
        Using excelEngine As ExcelEngine = New ExcelEngine()
            Dim application As IApplication = excelEngine.Excel
            application.DefaultVersion = ExcelVersion.Excel2013
            Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
            Dim worksheet As IWorksheet = workbook.Worksheets(0)
            'Map column headers in worksheet with class properties.
            Dim mappingProperties As Dictionary(Of String, String) = New Dictionary(Of
String, String) ()
            mappingProperties.Add("Customer ID", "CustId")
            mappingProperties.Add("Customer Name", "CustName")
            mappingProperties.Add("Customer Age", "CustAge")
            mappingProperties.Add("Order ID", "CustOrder.Order_Id")
            mappingProperties.Add("Order Price", "CustOrder.Price")
            'Export worksheet data into nested class Objects.
            Dim nestedClassObjects As List(Of Customer) = worksheet.ExportData(Of
Customer)(1, 1, 10, 5, mappingProperties)
            workbook.SaveAs("NestedClassObjects.xlsx")
        End Using
    End Sub
End Class

```

```

End Using
End Sub
End Class
'Customer details class
Public Partial Class Customer
Public Property CustId As Integer
Public Property CustName As String
Public Property CustAge As Integer
Public Property CustOrder As Order
Public Sub New()
End Sub
End Class
'Order details class
Public Partial Class Order
Public Property Order_Id As Integer
Public Property Price As Double
Public Sub New()
End Sub
End Class
End Namespace

```

UWP

```

using Syncfusion.XlsIO;
using System.Collections.Generic;
namespace ImportFromNestedCollection
{
public sealed partial class MainPage : Page
{
public MainPage()
{
this.InitializeComponent();
}
//Button click to Export data from worksheet to nested class objects.
private async void btnGenerateExcel_Click(object sender, RoutedEventArgs e)
{
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
//Map column headers in worksheet with class properties.
Dictionary<string, string> mappingProperties = new Dictionary<string,
string>();
mappingProperties.Add("Customer ID", "CustId");
mappingProperties.Add("Customer Name", "CustName");
mappingProperties.Add("Customer Age", "CustAge");
mappingProperties.Add("Order ID", "CustOrder.Order_Id");
mappingProperties.Add("Order Price", "CustOrder.Price");
//Export worksheet data into nested class Objects.
List<Customer> nestedClassObjects = worksheet.ExportData<Customer>(1, 1, 10,
5, mappingProperties);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
}
}
}

```

```

savePicker.SuggestedFileName = "NestedClassObjects";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
}
}
//Customer details class
public partial class Customer
{
public int CustId { get; set; }
public string CustName { get; set; }
public int CustAge { get; set; }
public Order CustOrder { get; set; }
public Customer()
{
}
}
//Order details class
public partial class Order
{
public int Order_Id { get; set; }
public double Price { get; set; }
public Order()
{
}
}
}
}

```

ASP.NET CORE

```

using Syncfusion.XlsIO;
using System.Collections.Generic;
namespace ImportFromNestedCollection
{
class Program
{
static void Main(string[] args)
{
ExportData();
}
//Main method to Export data from worksheet to nested class objects.
private static void ExportData()
{
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
//Map column headers in worksheet with class properties.
Dictionary<string, string> mappingProperties = new Dictionary<string,
string>();

```

```

mappingProperties.Add("Customer ID", "CustId");
mappingProperties.Add("Customer Name", "CustName");
mappingProperties.Add("Customer Age", "CustAge");
mappingProperties.Add("Order ID", "CustOrder.Order_Id");
mappingProperties.Add("Order Price", "CustOrder.Price");
//Export worksheet data into nested class Objects.
List<Customer> nestedClassObjects = worksheet.ExportData<Customer>(1, 1, 10,
5, mappingProperties);
//Saving the workbook as stream
FileStream stream = new FileStream("NestedClassObjects.xlsx",
    FileMode.Create, FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}
}
}
//Customer details class
public partial class Customer
{
    public int CustId { get; set; }
    public string CustName { get; set; }
    public int CustAge { get; set; }
    public Order CustOrder { get; set; }
    public Customer()
    {
    }
}
//Order details class
public partial class Order
{
    public int Order_Id { get; set; }
    public double Price { get; set; }
    public Order()
    {
    }
}
}

```

XAMARIN

```

using Syncfusion.XlsIO;
using System.Collections.Generic;
namespace ImportFromNestedCollection
{
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            this.InitializeComponent();
        }
        //Button click to Export data from worksheet to nested class objects.
        internal void OnButtonClicked(object sender, EventArgs e)
        {
            using (ExcelEngine excelEngine = new ExcelEngine())
            {
                IApplication application = excelEngine.Excel;
            }
        }
    }
}

```



```

application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
//Map column headers in worksheet with class properties.
Dictionary<string, string> mappingProperties = new Dictionary<string,
string>();
mappingProperties.Add("Customer ID", "CustId");
mappingProperties.Add("Customer Name", "CustName");
mappingProperties.Add("Customer Age", "CustAge");
mappingProperties.Add("Order ID", "CustOrder.Order_Id");
mappingProperties.Add("Order Price", "CustOrder.Price");
//Export worksheet data into nested class Objects.
List<Customer> nestedClassObjects = worksheet.ExportData<Customer>(1, 1, 10,
5, mappingProperties);
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Nested
ClassObjects.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("NestedClassObjects
.xlsx", "application/msexcel", stream);
}
}
}
}
//Customer details class
public partial class Customer
{
public int CustId { get; set; }
public string CustName { get; set; }
public int CustAge { get; set; }
public Order CustOrder { get; set; }
public Customer()
{
}
}
//Order details class
public partial class Order
{
public int Order_Id { get; set; }
public double Price { get; set; }
public Order()
{
}
}
}
}

```

Importing Data from Microsoft Grid Controls to Worksheet

XlsIO provides support to import data from various Microsoft grid controls with its cell formatting. The supported grid controls are:

- DataGridView
- GridView
- DataGridView

DataGrid

Imports data from Microsoft DataGrid control with its header and cell formatting to Excel worksheet. The following code illustrates how to import data from Microsoft DataGrid control to worksheet.

Note: GetDataTable() method returns DataTable of applicable data to import.

C#

```
//Initialize DataGrid control
DataGrid dataGrid = new DataGrid();
dataGrid.DataSource = GetDataTable();
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create();
IWorksheet worksheet = workbook.Worksheets[0];
//Import data from DataGrid control
worksheet.ImportDataGrid(dataGrid, 1, 1, true, true);
workbook.SaveAs("Output.xlsx");
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
'Initialize DataGrid control
Dim dataGrid As DataGrid = New DataGrid()
dataGrid.DataSource = GetDataTable()
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create()
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Import data from DataGrid control
worksheet.ImportDataGrid(dataGrid, 1, 1, True, True)
workbook.SaveAs("Output.xlsx")
workbook.Close()
excelEngine.Dispose()
```

UWP

```
//XlsIO supports importing of data from data grid to worksheet in Windows Forms and WPF platforms alone.
```

ASP.NET CORE

```
//XlsIO supports importing of data from data grid to worksheet in Windows Forms and WPF platforms alone.
```

XAMARIN

```
//XlsIO supports importing of data from data grid to worksheet in Windows Forms and WPF platforms alone.
```

GridView

Imports data from Microsoft GridView control with its header and cell formatting to Excel worksheet. The following code illustrates how to import data from Microsoft GridView control to worksheet.

C#

```
//Initialize GridView control
GridView gridView = new GridView();
gridView.DataSource = GetDataTable();
gridView.DataBind();
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create();
IWorksheet worksheet = workbook.Worksheets[0];
//Import data from GridView control
worksheet.ImportGridView(gridView, 1, 1, true, true);
workbook.SaveAs("Output.xlsx");
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
'Initialize GridView control
Dim gridView As GridView = New GridView ()
gridView.DataSource = GetDataTable()
gridView.DataBind()
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create()
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Import data from GridView control
worksheet.ImportGridView(gridView, 1, 1, True, True)
workbook.SaveAs("Output.xlsx")
workbook.Close()
excelEngine.Dispose()
```

UWP

```
//XlsIO supports importing of data from data view to worksheet in Windows Forms and WPF platforms alone.
```

ASP.NET CORE

```
//XlsIO supports importing of data from data view to worksheet in Windows Forms and WPF platforms alone.
```

XAMARIN

```
//XlsIO supports importing of data from data view to worksheet in Windows Forms and WPF platforms alone.
```

DataGridView

Imports data from Microsoft DataGridView control with its header and cell formatting to Excel worksheet. In addition, this API imports sorted data applied in the control. The following code illustrates how to import data from Microsoft DataGridView control to worksheet.

C#

```
//Initialize DataGridView control
DataGridView dataGridView = new DataGridView();
dataGridView.DataSource = GetDataTable();
//Apply sorting.
dataGridView.Sort(dataGridView.Columns[1],
System.ComponentModel.ListSortDirection.Ascending);
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create();
IWorksheet worksheet = workbook.Worksheets[0];
//Import data from DataGridView control
worksheet.ImportDataGridView(dataGridView, 1, 1, true, true);
workbook.SaveAs("Output.xlsx");
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
'Initialize DataGridView control
Dim dataGridView As DataGridView = New DataGridView()
dataGridView.DataSource = GetDataTable()
'Apply sorting.
dataGridView.Sort(dataGridView.Columns(1),
System.ComponentModel.ListSortDirection.Ascending)
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create()
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Import data from DataGridView control
worksheet.ImportDataGridView(dataGridView, 1, 1, True, True)
workbook.SaveAs("Output.xlsx")
workbook.Close()
excelEngine.Dispose()
```

UWP

```
//XlsIO supports importing of data from data grid view to worksheet in  
Windows Forms and WPF platforms alone.
```

ASP.NET CORE

```
//XlsIO supports importing of data from data grid view to worksheet in  
Windows Forms and WPF platforms alone.
```

XAMARIN

```
//XlsIO supports importing of data from data grid view to worksheet in  
Windows Forms and WPF platforms alone.
```

Working with Formulas

[Formulas](#) are entries in Excel that have equations, by which values are calculated. A typical formula might contain cell references, constants, and even functions.

Enable and Disable Calculation

To perform calculation in an Excel workbook, it is recommended to invoke **EnableSheetCalculations** method of **IWorksheet**. Enabling this method will initialize [CalcEngine](#) objects and retrieves calculated values of formulas in a worksheet.

The following code sample illustrates on how to enable worksheet formula calculations.

C#

```
IWorksheet sheet = workbook.Worksheets[0];  
//Formula calculation is enabled for the sheet  
sheet.EnableSheetCalculations();
```

VB.NET

```
Dim sheet As IWorksheet = workbook.Worksheets(0)  
'Formula calculation is enabled for the sheet  
sheet.EnableSheetCalculations()
```

UWP

```
IWorksheet sheet = workbook.Worksheets[0];  
//Formula calculation is enabled for the sheet  
sheet.EnableSheetCalculations();
```

ASP.NET CORE

```
IWorksheet sheet = workbook.Worksheets[0];  
//Formula calculation is enabled for the sheet  
sheet.EnableSheetCalculations();
```

XAMARIN

```
IWorksheet sheet = workbook.Worksheets[0];  
//Formula calculation is enabled for the sheet  
sheet.EnableSheetCalculations();
```

On completion of worksheet calculation, it is recommended to invoke **DisableSheetCalculations** method of **IWorksheet**. This will dispose all the [CalcEngine](#) objects.

The following code sample illustrates on how to disable worksheet formula calculations.

C#

```
IWorksheet sheet = workbook.Worksheets[0];  
//Formula calculation is disabled for the sheet  
sheet.DisableSheetCalculations();
```

VB.NET

```
Dim sheet As IWorksheet = workbook.Worksheets(0)  
'Formula calculation is disabled for the sheet  
sheet.DisableSheetCalculations()
```

UWP

```
IWorksheet sheet = workbook.Worksheets[0];  
//Formula calculation is disabled for the sheet  
sheet.DisableSheetCalculations();
```

ASP.NET CORE

```
IWorksheet sheet = workbook.Worksheets[0];  
//Formula calculation is disabled for the sheet  
sheet.DisableSheetCalculations();
```

XAMARIN

```
IWorksheet sheet = workbook.Worksheets[0];  
//Formula calculation is disabled for the sheet  
sheet.DisableSheetCalculations();
```

Writing a Formula

In a worksheet, formulas can be entered by using the **Formula** property of **IRange** instance. Following code example illustrates on how to write a formula.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())  
{  
    IApplication application = excelEngine.Excel;  
    application.DefaultVersion = ExcelVersion.Excel2013;  
    IWorkbook workbook = application.Workbooks.Create(1);  
    IWorksheet sheet = workbook.Worksheets[0];  
    //Setting values to the cells  
    sheet.Range["A1"].Number = 10;  
    sheet.Range["B1"].Number = 10;  
    //Setting formula in the cell  
    sheet.Range["C1"].Formula = "=SUM(A1,B1)";  
    workbook.SaveAs("Formula.xlsx");  
}
```

```
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Setting values to the cells
sheet.Range("A1").Number = 10
sheet.Range("B1").Number = 10
'Setting formula for the range
sheet.Range("C1").Formula = "=SUM(A1,B1)"
workbook.SaveAs("Formula.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Setting values to the cells
    sheet.Range["A1"].Number = 10;
    sheet.Range["B1"].Number = 10;
    //Setting formula in the cell
    sheet.Range["C1"].Formula = "=SUM(A1,B1)";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Formula";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Setting values to the cells
    sheet.Range["A1"].Number = 10;
    sheet.Range["B1"].Number = 10;
    //Setting formula in the cell
    sheet.Range["C1"].Formula = "=SUM(A1,B1)";
}
```

```
//Saving the workbook as stream
FileStream stream = new FileStream("Formula.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Setting values to the cells
    sheet.Range["A1"].Number = 10;
    sheet.Range["B1"].Number = 10;
    //Setting formula in the cell
    sheet.Range["C1"].Formula = "=SUM(A1,B1)";
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Formul
            a.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Formula.xlsx",
            "application/msexcel", stream);
    }
}
```

Formula with Cross-sheet References

XlsIO supports using formulas across worksheets. The following code shows how to apply formula with cross-sheet references.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create();
    IWorksheet sheet = workbook.Worksheets[0];
    //Setting formula for the range with cross-sheet reference
    sheet.Range["C2"].Formula = "=SUM(Sheet2!B2, Sheet1!A2)";
}
```



```
workbook.SaveAs("Formula.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create()
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Setting formula for the range with cross-sheet reference
sheet.Range("C2").Formula = "=SUM(Sheet2!B2,Sheet1!A2)"
workbook.SaveAs("Formula.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create();
IWorksheet sheet = workbook.Worksheets[0];
//Setting formula for the range with cross-sheet reference
sheet.Range["C2"].Formula = "=SUM(Sheet2!B2,Sheet1!A2)";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Formula";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create();
IWorksheet sheet = workbook.Worksheets[0];
//Setting formula for the range with cross-sheet reference
sheet.Range["C2"].Formula = "=SUM(Sheet2!B2,Sheet1!A2)";
//Saving the workbook as stream
FileStream stream = new FileStream("Formula.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}
```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create();
    IWorksheet sheet = workbook.Worksheets[0];
    //Setting formula for the range with cross-sheet reference
    sheet.Range["C2"].Formula = "=SUM(Sheet2!B2,Sheet1!A2)";
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Formul
a.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Formula.xlsx",
"application/msexcel", stream);
    }
}

```

Reading a Formula

Formulas are string values which can be accessed using **Formula** property of **IRange**. If a cell has formula, the **Value** property of **IRange** will also return the formula as string.

The following code shows how to read a formula.

C#

```

//Returns the formula in C1 style notation
string formula = sheet["C1"].Formula;

```

VB.NET

```

'Returns the formula in C1 style notation
Dim formula as String = sheet("C1").Formula

```

UWP

```

//Returns the formula in C1 style notation
string formula = sheet["C1"].Formula;

```

ASP.NET CORE

```

//Returns the formula in C1 style notation

```

```
string formula = sheet["C1"].Formula;
```

XAMARIN

```
//Returns the formula in C1 style notation
string formula = sheet["C1"].Formula;
```

Accessing a Calculated value

To evaluate formula, it is must to [enable sheet calculation](#) in prior. After enabling the sheet calculation, the formula can be evaluated using **CalculatedValue** of **IRange**, which returns a string value.

The following code shows how to access a calculated value.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    sheet.EnableSheetCalculations();
    //Returns the calculated value of a formula using the most current inputs
    string calculatedValue = sheet["A1"].CalculatedValue;
    sheet.DisableSheetCalculations();
    workbook.SaveAs("Formula.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
    ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
sheet.EnableSheetCalculations()
'Returns the calculated value of a formula using the most current inputs
Dim calculatedValue As String = sheet("C1").CalculatedValue
sheet.DisableSheetCalculations()
workbook.SaveAs("Formula.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
}
```

```

StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
sheet.EnableSheetCalculations();
//Returns the calculated value of a formula using the most current inputs
string calculatedValue = sheet["C1"].CalculatedValue;
sheet.DisableSheetCalculations();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Formula";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
sheet.EnableSheetCalculations();
//Returns the calculated value of a formula using the most current inputs
string calculatedValue = sheet["C1"].CalculatedValue;
sheet.DisableSheetCalculations();
//Saving the workbook as stream
FileStream stream = new FileStream("Formula.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
ple.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);

```

```

IWorksheet sheet = workbook.Worksheets[0];
sheet.EnableSheetCalculations();
//Returns the calculated value of a formula using the most current inputs
string calculatedValue = sheet["C1"].CalculatedValue;
sheet.DisableSheetCalculations();
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Formul
a.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Formula.xlsx",
"application/msexcel", stream);
}
}

```

Apart from **CalculatedValue** property, the evaluated values can be accessed as **bool**, **DateTime** and **double** data types. To obtain updated values of these types, **CalculatedValue** property must be called in prior.

To know more about evaluated values, please refer **IRange** in API section.

The following code shows how to access calculated values in different types.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
//Previous Value '2'
sheet["E1"].Number = 3;
sheet.EnableSheetCalculations();
//It has formula 'ISEVEN(E1)'
//Returns the calculated value of a formula as Boolean
bool B1_PreviousValue = sheet["B1"].FormulaBoolValue;
string value = sheet.Range["B1"].CalculatedValue;
bool B1_LatestValue = sheet["B1"].FormulaBoolValue;
//It has formula 'TODAY()'
//Returns the calculated value of a formula as DateTime
DateTime C1_PreviousValue = sheet["C1"].FormulaDateTime;
value = sheet.Range["C1"].CalculatedValue;
DateTime C1_LatestValue = sheet["C1"].FormulaDateTime;
//It has formula '=E1'
}

```

```
//Returns the calculated value of a formula as double
double D1_PreviousValue = sheet["D1"].FormulaNumberValue;
value = sheet.Range["D1"].CalculatedValue;
double D1_LatestValue = sheet["D1"].FormulaNumberValue;
sheet.DisableSheetCalculations();
workbook.SaveAs("Formula.xlsx");
}

//Output
//B1_PreviousValue - true                                     B1_LatestValue - false
//C1_PreviousValue - {3/27/2018 12:00:00 AM}   C1_LatestValue - {3/28/2018
12:00:00 AM}
//D1_PreviousValue - 2.0                                   D1_LatestValue - 3.0
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Previous Value '2'
sheet("E1").Number = 3
sheet.EnableSheetCalculations()
'It has formula 'ISEVEN(E1) '
'Returns the calculated value of a formula as Boolean
Dim B1_PreviousValue As Boolean = sheet("B1").FormulaBoolValue
Dim value As String = sheet.Range("B1").CalculatedValue
Dim B1_LatestValue As Boolean = sheet("B1").FormulaBoolValue
'It has formula 'TODAY() '
'Returns the calculated value of a formula as DateTime
Dim C1_PreviousValue As DateTime = sheet("C1").FormulaDateTime
value = sheet.Range("C1").CalculatedValue
Dim C1_LatestValue As DateTime = sheet("C1").FormulaDateTime
'It has formula '=E1 '
'Returns the calculated value of a formula as double
Dim D1_PreviousValue As Double = sheet("D1").FormulaNumberValue
value = sheet.Range("D1").CalculatedValue
Dim D1_LatestValue As Double = sheet("D1").FormulaNumberValue
sheet.DisableSheetCalculations()
workbook.SaveAs("Formula.xlsx")
End Using

'Output
'B1_PreviousValue - true                                     B1_LatestValue - false
'C1_PreviousValue - {3/27/2018 12:00:00 AM}   C1_LatestValue - {3/28/2018
12:00:00 AM}
'D1_PreviousValue - 2.0                                   D1_LatestValue - 3.0
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
```

```

openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
//Previous Value '2'
sheet["E1"].Number = 3;
sheet.EnableSheetCalculations();
//It has formula 'ISEVEN(E1)'
//Returns the calculated value of a formula as Boolean
bool B1_PreviousValue = sheet["B1"].FormulaBoolValue;
string value = sheet.Range["B1"].CalculatedValue;
bool B1_LatestValue = sheet["B1"].FormulaBoolValue;
//It has formula 'TODAY()'
//Returns the calculated value of a formula as DateTime
DateTime C1_PreviousValue = sheet["C1"].FormulaDateTime;
value = sheet.Range["C1"].CalculatedValue;
DateTime C1_LatestValue = sheet["C1"].FormulaDateTime;
//It has formula '=E1'
//Returns the calculated value of a formula as double
double D1_PreviousValue = sheet["D1"].FormulaNumberValue;
value = sheet.Range["D1"].CalculatedValue;
double D1_LatestValue = sheet["D1"].FormulaNumberValue;
sheet.DisableSheetCalculations();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Formula";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
//Output
//B1_PreviousValue - true                                     B1_LatestValue - false
//C1_PreviousValue - {3/27/2018 12:00:00 AM}   C1_LatestValue - {3/28/2018
12:00:00 AM}
//D1_PreviousValue - 2.0                                   D1_LatestValue - 3.0

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Previous Value '2'

```

```

sheet["E1"].Number = 3;
sheet.EnableSheetCalculations();
//It has formula 'ISEVEN(E1)'
//Returns the calculated value of a formula as Boolean
bool B1_PreviousValue = sheet["B1"].FormulaBoolValue;
string value = sheet.Range["B1"].CalculatedValue;
bool B1_LatestValue = sheet["B1"].FormulaBoolValue;
//It has formula 'TODAY()'
//Returns the calculated value of a formula as DateTime
DateTime C1_PreviousValue = sheet["C1"].FormulaDateTime;
value = sheet.Range["C1"].CalculatedValue;
DateTime C1_LatestValue = sheet["C1"].FormulaDateTime;
//It has formula '=E1'
//Returns the calculated value of a formula as double
double D1_PreviousValue = sheet["D1"].FormulaNumberValue;
value = sheet.Range["D1"].CalculatedValue;
double D1_LatestValue = sheet["D1"].FormulaNumberValue;
sheet.DisableSheetCalculations();
//Saving the workbook as stream
FileStream stream = new FileStream("Formula.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}
//Output
//B1_PreviousValue - true                                B1_LatestValue - false
//C1_PreviousValue - {3/27/2018 12:00:00 AM}  C1_LatestValue - {3/28/2018
12:00:00 AM}
//D1_PreviousValue - 2.0                                D1_LatestValue - 3.0

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    // "App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Previous Value '2'
    sheet["E1"].Number = 3;
    sheet.EnableSheetCalculations();
    //It has formula 'ISEVEN(E1)'
    //Returns the calculated value of a formula as Boolean
    bool B1_PreviousValue = sheet["B1"].FormulaBoolValue;
    string value = sheet.Range["B1"].CalculatedValue;
    bool B1_LatestValue = sheet["B1"].FormulaBoolValue;
    //It has formula 'TODAY()'
    //Returns the calculated value of a formula as DateTime
    DateTime C1_PreviousValue = sheet["C1"].FormulaDateTime;
    value = sheet.Range["C1"].CalculatedValue;
}

```



```

DateTime C1_LatestValue = sheet["C1"].FormulaDateTime;
//It has formula '=E1'
//Returns the calculated value of a formula as double
double D1_PreviousValue = sheet["D1"].FormulaNumberValue;
value = sheet.Range["D1"].CalculatedValue;
double D1_LatestValue = sheet["D1"].FormulaNumberValue;
sheet.DisableSheetCalculations();
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Formul
a.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Formula.xlsx",
"application/msexcel", stream);
}
}
//Output
//B1_PreviousValue - true                                     B1_LatestValue - false
//C1_PreviousValue - {3/27/2018 12:00:00 AM}   C1_LatestValue - {3/28/2018
12:00:00 AM}
//D1_PreviousValue - 2.0                                   D1_LatestValue - 3.0

```

Note: Calculated value for external reference formulas can be evaluated in XlsIO.

Applying Argument Separators Based on Cultures

Formula separators vary for different cultures, and exceptions can be thrown in such cases. This can be overcome by setting the separators by using **SetSeparators** method of **IWorkbook**.

Following code illustrates on how to change the formula separators.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Setting the argument separator
workbook.SetSeparators(';', ',', ' ');
workbook.SaveAs("Formula.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
'Setting the argument separator
workbook.SetSeparators(";", ",")
workbook.SaveAs("Formula.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Setting the argument separator
    workbook.SetSeparators(';', ',');
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Formula";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Setting the argument separator
    workbook.SetSeparators(';', ',');
    //Saving the workbook as stream
    FileStream stream = new FileStream("Formula.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Setting the argument separator
    workbook.SetSeparators(';', ',');
}

```

```
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Formul
a.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Formula.xlsx",
"application/msexcel", stream);
}
}
```

Array of Formula

Array formula is a special type of formula in Excel. It works with an array or series of data values, rather than a single data value which can be done through **FormulaArray** property of **IRange** instance.

Following code shows how an array of values from [Named Range](#) is used for computation.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Assign array formula
sheet.Range["A1:D1"].FormulaArray = "{1,2,3,4}";
//Adding a named range for the range A1 to D1
sheet.Names.Add("ArrayRange", sheet.Range["A1:D1"]);
//Assign formula array with named range
sheet.Range["A2:D2"].FormulaArray = "ArrayRange+100";
string fileName = "FormulaArray.xlsx";
workbook.SaveAs(fileName);
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Assign array formula
sheet.Range("A1:D1").FormulaArray = "{1,2,3,4}"
'Adding a named range for the range A1 to D1
sheet.Names.Add("ArrayRange", sheet.Range("A1:D1"))
```

```
'Assign formula array with named range
sheet.Range("A2:D2").FormulaArray = "ArrayRange+100"
workbook.SaveAs("FormulaArray.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Assign array formula
    sheet.Range["A1:D1"].FormulaArray = "{1,2,3,4}";
    //Adding a named range for the range A1 to D1
    sheet.Names.Add("ArrayRange", sheet.Range["A1:D1"]);
    //Assign formula array with named range
    sheet.Range["A2:D2"].FormulaArray = "ArrayRange+100";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "FormulaArray";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Assign array formula
    sheet.Range["A1:D1"].FormulaArray = "{1,2,3,4}";
    //Adding a named range for the range A1 to D1
    sheet.Names.Add("ArrayRange", sheet.Range["A1:D1"]);
    //Assign formula array with named range
    sheet.Range["A2:D2"].FormulaArray = "ArrayRange+100";
    //Saving the workbook as stream
    FileStream stream = new FileStream("FormulaArray.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
```

```

{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Assign array formula
    sheet.Range["A1:D1"].FormulaArray = "{1,2,3,4}";
    //Adding a named range for the range A1 to D1
    sheet.Names.Add("ArrayRange", sheet.Range["A1:D1"]);
    //Assign formula array with named range
    sheet.Range["A2:D2"].FormulaArray = "ArrayRange+100";
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Formul
aArray.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("FormulaArray.xlsx"
, "application/msexcel", stream);
    }
}

```

Incremental Formula

The relative cell references in the formulas are automatically incremented by 1, when you fill formulas down a column or across a row by enabling the `EnableIncrementalFormula` property of **IApplication** interface.

The below code snippet shows how to increment the cell references by 1 in the formulas.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Enables the incremental formula to updates the reference in cell
    application.EnableIncrementalFormula = true;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Formula are automatically increments by one for the range of cells
    sheet["A1:A5"].Formula = "=B1+C1";
    workbook.SaveAs("IncrementalFormula.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
'Enables the incremental formula to updates the reference in cell
application.EnableIncrementalFormula = True
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Formula are automatically increments by one for the range of cells
sheet("A1:A5").Formula = "=B1+C1"
workbook.SaveAs("IncrementalFormula.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Enables the incremental formula to updates the reference in cell
    application.EnableIncrementalFormula = true;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Formula are automatically increments by one for the range of cells
    sheet["A1:A5"].Formula = "=B1+C1";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "IncrementalFormula";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Enables the incremental formula to updates the reference in cell
    application.EnableIncrementalFormula = true;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Formula are automatically increments by one for the range of cells
    sheet["A1:A5"].Formula = "=B1+C1";
    //Saving the workbook as stream
    FileStream stream = new FileStream("IncrementalFormula.xlsx",
    FileMode.Create, FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Enables the incremental formula to updates the reference in cell
    application.EnableIncrementalFormula = true;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Formula are automatically increments by one for the range of cells
    sheet["A1:A5"].Formula = "=B1+C1";
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("IncrementalFormula.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("IncrementalFormula.xlsx", "application/msexcel", stream);
    }
}

```

External Formula

XlsIO supports to write and preserve external formula.

External formula is the one which refers to a cell or a range of cells or a defined named range from outside the current worksheet/workbook. The main benefit of using an external reference is that whenever the referenced cell(s) in another worksheet changes, the value returned by the external cell reference is automatically updated.

Following code illustrates the insertion of a formula that refers to cell 'A1' in another workbook which is enclosed in a square bracket [One.xlsx].

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Write an external formula value
    sheet.Range["C1"].Formula = "[One.xlsx]Sheet1!$A$1*5";
    workbook.SaveAs("Formula.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Write an external formula value
sheet.Range("C1").Formula = "[One.xlsx]Sheet1!$A$1*5"
workbook.SaveAs("Formula.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Write an external formula value
    sheet.Range["C1"].Formula = "[One.xlsx]Sheet1!$A$1*5";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Formula";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Write an external formula value
    sheet.Range["C1"].Formula = "[One.xlsx]Sheet1!$A$1*5";
    //Saving the workbook as stream
    FileStream stream = new FileStream("Formula.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())

```



```

{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Write an external formula value
    sheet.Range["C1"].Formula = "[One.xlsx]Sheet1!$A$1*5";
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Formul
a.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Formula.xlsx",
"application/msexcel", stream);
    }
}

```

Note: Links are updated automatically in Microsoft Excel to view the result for the preceding code.

Calculated Column

XlsIO supports to create, access and modify [calculated column](#) in a table. When you enter a formula in a table column, Excel creates a calculated column. This column uses a single formula that's automatically extended to additional rows in the column and adjusted for each row. You just enter a formula once, and Excel immediately fills it down to create the calculated column.

Also, XlsIO supports [structured reference](#) in calculated column in table from Excel 2013.

The following code snippet illustrates how to create a calculated column.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create Table with data in the given range
    IListObject table = worksheet.ListObjects.Create("Table1",
worksheet["A1:D3"]);
    //Create data
    worksheet[1, 1].Text = "Products";
    worksheet[1, 2].Text = "Rate";
    worksheet[1, 3].Text = "Quantity";
    worksheet[1, 4].Text = "Total";
}

```

```

worksheet[2, 1].Text = "Item1";
worksheet[2, 2].Number = 200;
worksheet[2, 3].Number = 2;
worksheet[3, 1].Text = "Item2";
worksheet[3, 2].Number = 200;
worksheet[3, 3].Number = 2;
//Set table formula
table.Columns[3].CalculatedFormula = "SUM(20,[Rate]*[Quantity])";
string fileName = "Output.xlsx";
workbook.SaveAs(fileName);
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Create Table with data in the given range
Dim table As IListObject = worksheet.ListObjects.Create("Table1",
worksheet("A1:D3"))
'Create data
worksheet(1, 1).Text = "Products"
worksheet(1, 2).Text = "Rate"
worksheet(1, 3).Text = "Quantity"
worksheet(1, 4).Text = "Total"
worksheet(2, 1).Text = "Item1"
worksheet(2, 2).Number = 200
worksheet(2, 3).Number = 2
worksheet(3, 1).Text = "Item2"
worksheet(3, 2).Number = 200
worksheet(3, 3).Number = 2
'Set table formula
table.Columns(3).CalculatedFormula = "SUM(20,[Rate]*[Quantity])"
Dim fileName As String = "Output.xlsx"
workbook.SaveAs(fileName)
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Create Table with data in the given range
IListObject table = worksheet.ListObjects.Create("Table1",
worksheet["A1:D3"]);
//Create data
worksheet[1, 1].Text = "Products";
worksheet[1, 2].Text = "Rate";
worksheet[1, 3].Text = "Quantity";
worksheet[1, 4].Text = "Total";
worksheet[2, 1].Text = "Item1";

```

```

worksheet[2, 2].Number = 200;
worksheet[2, 3].Number = 2;
worksheet[3, 1].Text = "Item2";
worksheet[3, 2].Number = 200;
worksheet[3, 3].Number = 2;
//Set table formula
table.Columns[3].CalculatedFormula = "SUM(20,[Rate]*[Quantity])";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create Table with data in the given range
    IListObject table = worksheet.ListObjects.Create("Table1",
    worksheet["A1:D3"]);
    //Create data
    worksheet[1, 1].Text = "Products";
    worksheet[1, 2].Text = "Rate";
    worksheet[1, 3].Text = "Quantity";
    worksheet[1, 4].Text = "Total";
    worksheet[2, 1].Text = "Item1";
    worksheet[2, 2].Number = 200;
    worksheet[2, 3].Number = 2;
    worksheet[3, 1].Text = "Item2";
    worksheet[3, 2].Number = 200;
    worksheet[3, 3].Number = 2;
    //Set table formula
    table.Columns[3].CalculatedFormula = "SUM(20,[Rate]*[Quantity])";
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;

```

```

IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Create Table with data in the given range
IListObject table = worksheet.ListObjects.Create("Table1",
worksheet["A1:D3"]);
//Create data
worksheet[1, 1].Text = "Products";
worksheet[1, 2].Text = "Rate";
worksheet[1, 3].Text = "Quantity";
worksheet[1, 4].Text = "Total";
worksheet[2, 1].Text = "Item1";
worksheet[2, 2].Number = 200;
worksheet[2, 3].Number = 2;
worksheet[3, 1].Text = "Item2";
worksheet[3, 2].Number = 200;
worksheet[3, 3].Number = 2;
//Set table formula
table.Columns[3].CalculatedFormula = "SUM(20,[Rate]*[Quantity])";
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Supported Functions

XlsIO supports all the formulas supported by Excel. Whereas, below is the list of functions that XlsIO performs calculation and returns a calculated value.

Functions	Description
ABS	Returns the absolute value of a number
ACOS	Returns the arccosine of a number
ACOSH	Returns the inverse hyperbolic cosine of a number
ADDRESS	Returns a reference as text to a single cell in a worksheet

AND	Returns TRUE if all of its arguments are TRUE
AREAS	Returns the number of areas in a reference
ASC	Changes full-width (double-byte) English letters or katakana within a character string to half-width (single-byte) characters
ASIN	Returns the arcsine of a number
ASINH	Returns the inverse hyperbolic sine of a number
ATAN	Returns the arctangent of a number
ATAN2	Returns the arctangent from x- and y-coordinates
ATANH	Returns the inverse hyperbolic tangent of a number
AVEDEV	Returns the average of the absolute deviations of data points from their mean
AVERAGE	Returns the average of its arguments
AVERAGEA	Returns the average of its arguments, including numbers, text, and logical values
AVERAGEIF	Returns the average (arithmetic mean) of all the cells in a range that meet a given criterion
AVERAGEIFS	Returns the average (arithmetic mean) of all cells that meet multiple criteria
BESSELI	Returns the modified Bessel function $I_n(x)$
BESSELJ	Returns the Bessel function $J_n(x)$
BESSELK	Returns the modified Bessel function $K_n(x)$
BESSELY	Returns the Bessel function $Y_n(x)$
BIN2DEC	Converts a binary number to decimal
BIN2HEX	Converts a binary number to hexadecimal
BIN2OCT	Converts a binary number to octal
BINOMDIST	Returns the individual term binomial distribution probability
CEILING	Rounds a number to the nearest integer or to the nearest multiple of significance

CELL	Returns information about the formatting, location, or contents of a cell
CHAR	Returns the character specified by the code number
CHIDIST	Returns the one-tailed probability of the chi-squared distribution
CHIINV	Returns the inverse of the one-tailed probability of the chi-squared distribution
CHITEST	Returns the test for independence
CHOOSE	Chooses a value from a list of values
CLEAN	Removes all non-printable characters from text
CODE	Returns a numeric code for the first character in a text string
COLUMN	Returns the column number of a reference
COLUMNS	Returns the number of columns in a reference
COMBIN	Returns the number of combinations for a given number of objects
COMPLEX	Converts real and imaginary coefficients into a complex number
CONCAT	Combines the text from multiple ranges and/or strings
CONCATENATE	Joins several text items into one text item
CONFIDENCE	Returns the confidence interval for a population mean
CONVERT	Converts a number from one measurement system to another
CORREL	Returns the correlation coefficient between two data sets
COS	Returns the cosine of a number
COSH	Returns the hyperbolic cosine of a number
COUNT	Counts how many numbers are in the list of arguments
COUNTA	Counts how many values are in the list of arguments
COUNTBLANK	Counts the number of blank cells within a range
COUNTIF	Counts the number of non-blank cells within a range that meet the given criteria

COVAR	Returns covariance, the average of the products of paired deviations
CRITBINOM	Returns the smallest value for which the cumulative binomial distribution is less than or equal to a criterion value
CUMIPMT	Returns the cumulative interest paid between two periods
CUMPRINC	Returns the cumulative principal paid on a loan between two periods
DATE	Returns the serial number of a particular date
DATEVALUE	Converts a date in the form of text to a serial number
DAY	Converts a serial number to a day of the month
DAYS360	Calculates the number of days between two dates based on a 360-day year
DB	Returns the depreciation of an asset for a specified period by using the fixed-declining balance method
DDB	Returns the depreciation of an asset for a specified period by using the double-declining balance method or some other method that you specify
DEC2BIN	Converts a decimal number to binary
DECHEX	Converts a decimal number to hexadecimal
DEC2OCT	Converts a decimal number to octal
DEGREES	Converts radians to degrees
DELTA	Tests whether two values are equal
DEVSQ	Returns the sum of squares of deviations
DISC	Returns the discount rate for a security
DOLLAR	Converts a number to text, using the \$ (dollar) currency format
DOLLARDE	Converts a dollar price, expressed as a fraction, into a dollar price, expressed as a decimal number
DOLLARFR	Converts a dollar price, expressed as a decimal number, into a dollar price, expressed as a fraction

DURATION	Returns the annual duration of a security with periodic interest payments
EDATE	Returns the serial number of the date that is the indicated number of months before or after the start date
EFFECT	Returns the effective annual interest rate
EOMONTH	Returns the serial number of the last day of the month before or after a specified number of months
ERF	Returns the error function
ERFC	Returns the complementary error function
ERROR.TYPE	Returns a number corresponding to an error type
EVEN	Rounds a number up to the nearest even integer
EXACT	Checks to see if two text values are identical
EXP	Returns e raised to the power of a given number
EXPONDIST	Returns the exponential distribution
FACT	Returns the factorial of a number
FACTDOUBLE	Returns the double factorial of a number
FDIST	Returns the F probability distribution
FIND, FINDB	Finds one text value within another (case-sensitive)
FINV	Returns the inverse of the F probability distribution
FISHER	Returns the Fisher transformation
FISHER	Returns the inverse of the Fisher transformation
FIXED	Formats a number as text with a fixed number of decimals
FLOOR	Rounds a number down, toward zero
FORECAST	Returns a value along a linear trend
FV	Returns the future value of an investment

FVSCHEDULE	Returns the future value of an initial principal after applying a series of compound interest rates
GAMMADIST	Returns the gamma distribution
GAMMAINV	Returns the inverse of the gamma cumulative distribution
GAMMALIN	Returns the natural logarithm of the gamma function, $\hat{\Gamma}(x)$
GCD	Returns the greatest common divisor
GEOMEAN	Returns the geometric mean
GESTEP	Tests whether a number is greater than a threshold value
GROWTH	Returns values along an exponential trend
HARMEAN	Returns the harmonic mean
HEX2BIN	Converts a hexadecimal number to binary
HEX2DEC	Converts a hexadecimal number to decimal
HEX2OCT	Converts a hexadecimal number to octal
HLOOKUP	Looks in the top row of an array and returns the value of the indicated cell
HOUR	Converts a serial number to an hour
HYPERLINK	Creates a shortcut or jump that opens a document stored on a network server, an intranet, or the Internet
HYPGEOMDIST	Returns the hypergeometric distribution
IF	Specifies a logical test to perform
IFERROR	Returns a specified value if a formula evaluates to an error.
IFS	Checks whether one or more conditions are met and returns a value that corresponds to the first TRUE condition
IMABS	Returns the absolute value (modulus) of a complex number
IMAGINARY	Returns the imaginary coefficient of a complex number

IMARGUMENT	Returns the argument theta, an angle expressed in radians
IMCONJUGATE	Returns the complex conjugate of a complex number
IMCOS	Returns the cosine of a complex number
IMDIV	Returns the quotient of two complex numbers
IMEXP	Returns the exponential of a complex number
IMLN	Returns the natural logarithm of a complex number
IMLOG10	Returns the base-10 logarithm of a complex number
IMLOG2	Returns the base-2 logarithm of a complex number
IMPOWER	Returns a complex number raised to an integer power
IMPRODUCT	Returns the product of from 2 to 29 complex numbers
IMREAL	Returns the real coefficient of a complex number
IMSIN	Returns the sine of a complex number
IMSQRT	Returns the square root of a complex number
IMSUB	Returns the difference between two complex numbers
IMSUM	Returns the sum of complex numbers
INDEX	Uses an index to choose a value from a reference or array
INDIRECT	Returns a reference indicated by a text value
INFO	Returns information about the current operating environment
INT	Rounds a number down to the nearest integer
INTERCEPT	Returns the intercept of the linear regression line
INTRATE	Returns the interest rate for a fully invested security
IPMT	Returns the interest payment for an investment for a given period
IRR	Returns the internal rate of return for a series of cash flows

ISBLANK	Returns TRUE if the value is blank
ISERR	Returns TRUE if the value is any error value except #N/A
ISERROR	Returns TRUE if the value is any error value
ISEVEN	Returns TRUE if the number is even
ISLOGICAL	Returns TRUE if the value is a logical value
ISAN	Returns TRUE if the value is the #N/A error value
ISNONTEXT	Returns TRUE if the value is not text
ISNUMBER	Returns TRUE if the value is a number
ISODD	Returns TRUE if the number is odd
ISMPT	Calculates the interest paid during a specific period of an investment
ISREF	Returns TRUE if the value is a reference
ISTEXT	Returns TRUE if the value is text
KURT	Returns the kurtosis of a data set
LARGE	Returns the k-th largest value in a data set
LCM	Returns the least common multiple
LEFT, LEFTB	Returns the leftmost characters from a text value
LEN, LENB	Returns the number of characters in a text string
LN	Returns the natural logarithm of a number
LOG	Returns the logarithm of a number to a specified base
LOG10	Returns the base-10 logarithm of a number
LOGEST	Returns the parameters of an exponential trend
LOGINV	Returns the inverse of the log-normal distribution
LOGNORMDIST	Returns the cumulative log-normal distribution

LOOKUP	Looks up values in a vector or array
LOWER	Converts text to lowercase
MATCH	Looks up values in a reference or array
MAX	Returns the maximum value in a list of arguments
MAXA	Returns the maximum value in a list of arguments, including numbers, text, and logical values
MAXIFS	Returns the maximum value among cells specified by a given set of conditions or criteria
MDETERM	Returns the matrix determinant of an array
MEDIAN	Returns the median of the given numbers
MID, MIDB	Returns a specific number of characters from a text string starting at the position you specify
MIN	Returns the minimum value in a list of arguments
MINA	Returns the smallest value in a list of arguments, including numbers, text, and logical values
MINIFS	Returns the minimum value among cells specified by a given set of conditions or criteria
MINUTE	Converts a serial number to a minute
MINVERSE	Returns the matrix inverse of an array
MIRR	Returns the internal rate of return where positive and negative cash flows are financed at different rates
MMULT	Returns the matrix product of two arrays
MOD	Returns the remainder from division
MODE	Returns the most common value in a data set
MMONTH	Converts a serial number to a month
MROUND	Returns a number rounded to the desired multiple

MULTINOMINAL	Returns the multinomial of a set of numbers
N	Returns a value converted to a number
NA	Returns the error value #N/A
NEGBINOMDIST	Returns the negative binomial distribution
NETWORKDAYS	Returns the number of whole workdays between two dates
NORMDIST	Returns the normal cumulative distribution
NORMINV	Returns the inverse of the normal cumulative distribution
NORMSDIST	Returns the standard normal cumulative distribution
NORMSINV	Returns the inverse of the standard normal cumulative distribution
NOT	Reverses the logic of its argument
NOW	Returns the serial number of the current date and time
NPER	Returns the number of periods for an investment
NPV	Returns the net present value of an investment based on a series of periodic cash flows and a discount rate
OCT2BIN	Converts an octal number to binary
OCT2DEC	Converts an octal number to decimal
OCT2HEX	Converts an octal number to hexadecimal
ODD	Rounds a number up to the nearest odd integer
OFFSET	Returns a reference offset from a given reference
OR	Returns TRUE if any argument is TRUE
PEARSON	Returns the Pearson product moment correlation coefficient
PERCENTILE	Returns the k-th percentile of values in a range
PERCENTRANK	Returns the percentage rank of a value in a data set

PERMUT	Returns the number of permutations for a given number of objects
PI	Returns the value of pi
PMT	Returns the periodic payment for an annuity
POISSON	Returns the Poisson distribution
POWER	Returns the result of a number raised to a power
PPMT	Returns the payment on the principal for an investment for a given period
PROB	Returns the probability that values in a range are between two limits
PRODUCT	Multiplies its arguments
PROPER	Capitalizes the first letter in each word of a text value
PV	Returns the present value of an investment
QUARTILE	Returns the quartile of a data set
QUOTIENT	Returns the integer portion of a division
RADIANS	Converts degrees to radians
RAND	Returns a random number between 0 and 1
RANDBETWEEN	Returns a random number between the numbers you specify
RANK	Returns the rank of a number in a list of numbers
RATE	Returns the interest rate per period of an annuity
RECEIVED	Returns the amount received at maturity for a fully invested security
REPLACE, REPLACEB	Replaces characters within text
REPT	Repeats text a given number of times
RIGHT, RIGHTB	Returns the rightmost characters from a text value
ROMAN	Converts an Arabic numeral to roman, as text
ROUND	Rounds a number to a specified number of digits

ROUNDDOWN	Rounds a number down, toward zero
ROUNDUP	Rounds a number up, away from zero
ROW	Returns the row number of a reference
ROWS	Returns the number of rows in a reference
RSQ	Returns the square of the Pearson product moment correlation coefficient
SEARCH, SEARCHB	Finds one text value within another (not case-sensitive)
SECOND	Converts a serial number to a second
SERIESSUM	Returns the sum of a power series based on the formula
SIGN	Returns the sign of a number
SIN	Returns the sine of the given angle
SINH	Returns the hyperbolic sine of a number
SKEW	Returns the skewness of a distribution
SLN	Returns the straight-line depreciation of an asset for one period
SLOPE	Returns the slope of the linear regression line
SMALL	Returns the k-th smallest value in a data set
SQRT	Returns a positive square root
SQRTPI	Returns the square root of (number * pi)
STANDARDIZE	Returns a normalized value
STDEV	Estimates standard deviation based on a sample
STDEVA	Estimates standard deviation based on a sample, including numbers, text, and logical values
STDEVP	Calculates standard deviation based on the entire population
STDEVPA	Calculates standard deviation based on the entire population, including numbers, text, and logical values

STEYX	Returns the standard error of the predicted y-value for each x in the regression
SUBSTITUTE	Substitutes new text for old text in a text string
SUBTOTAL	Returns a subtotal in a list or database
SUM	Adds its arguments
SUMIF	Adds the cells specified by a given criteria
SUMPRODUCT	Returns the sum of the products of corresponding array components
SUMSQ	Returns the sum of the squares of the arguments
SUMX2MY2	Returns the sum of the difference of squares of corresponding values in two arrays
SUMX2PY2	Returns the sum of the sum of squares of corresponding values in two arrays
SUMXMY2	Returns the sum of squares of differences of corresponding values in two arrays
SWITCH	Evaluates an expression against a list of values and returns the result corresponding to the first matching value. If there is no match, an optional default value may be returned.
SYD	Returns the sum-of-years'digits depreciation of an asset for a specified period
T	Converts its arguments to text
TAN	Returns the tangent of a number
TANH	Returns the hyperbolic tangent of a number
TEXT	Formats a number and converts it to text
TEXTJOIN	Combines the text from multiple ranges and/or strings with a delimiter you specify between each text value that will be combined
TIME	Returns the serial number of a particular time
TIMEVALUE	Converts a time in the form of text to a serial number
TODAY	Returns the serial number of today's date
TRANSPOSE	Returns the transpose of an array

TRIM	Removes spaces from text
TRIMMEAN	Returns the mean of the interior of a data set
TRUNC	Truncates a number to an integer
TYPE	Returns a number indicating the data type of a value
UPPER	Converts text to uppercase
VALUE	Converts a text argument to a number
VAR	Estimates variance based on a sample
VARA	Estimates variance based on a sample, including numbers, text, and logical values
VARP	Calculates variance based on the entire population
VARPA	Calculates variance based on the entire population, including numbers, text, and logical values
VDB	Returns the depreciation of an asset for a specified or partial period by using a declining balance method
VLOOKUP	Looks in the first column of an array and moves across the row to return the value of a cell
WEEKDAY	Converts a serial number to a day of the week
WEEKNUM	Converts a serial number to a number representing where the week falls numerically with a year
WEIBULL	Returns the Weibull distribution
WORKDAY	Returns the serial number of the date before or after a specified number of workdays
XIRR	Returns the internal rate of return for a schedule of cash flows that is not necessarily periodic
YEAR	Converts a serial number to a year
YEARFRAC	Returns the year fraction representing the number of whole days between <i>startdate</i> and <i>enddate</i>

ZTEST	Returns the one-tailed probability-value of a z-test
FALSE	Returns the logical value FALSE
TRUE	Returns the logical value TRUE

Add-in Functions

Add-ins are mini-programs or custom functions that enhance the feature set of the Microsoft Excel application. These Add-ins can be accessed by registering it at first from Excel and refer it using XlsIO. For more details on adding AddIn functions, see [Add or remove Add-ins](#)

The following code illustrates on how to include and access Add-ins in XlsIO.

C#

```
//Step1: Create AddIn (AddIn.xlam)
//AddIn.xlam file has the below custom function
//Function AddInFunction(firstValue As Integer, secondValue As Integer) As Integer
//
//Dim result As Integer
//result = firstValue + secondValue
//AddInFunction = result
//
//End Function
//Step2: Register the AddIn in Excel by adding the above file to Excel
//Application by locating the .xlam file through the menu (Developer -> Add-
//ins -> Browse)
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    IAddInFunctions unknownFunctions = workbook.AddInFunctions;
    //Adding the XLAM file reference to AddIn functions
    //NOTE: The add-in name must be same as the function name
    unknownFunctions.Add(@"D:\AddIn.xlam", "AddInFunction");
    //Use the function. The expected result is 30
    sheet.Range["A3"].Formula = "AddInFunction(10,20)";
    string fileName = "AddIn.xlsx";
    workbook.Version = ExcelVersion.Excel2010;
    workbook.SaveAs(fileName);
}
```

VB.NET

```
'Step1: Create AddIn (AddIn.xlam)
'AddIn.xlam file has the below custom function
'Function AddInFunction(firstValue As Integer, secondValue As Integer) As Integer
'
'Dim result As Integer
```

```

'result = firstValue + secondValue
'AddInFunction = result
'
'End Function
'Step2: Register the AddIn in Excel by adding the above file to Excel
Application by locating the .xlam file through the menu (Developer -> Add-
ins -> Browse)
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim unknownFunctions As IAddInFunctions = workbook.AddInFunctions
'Adding the XLAM file reference to AddIn functions
'NOTE: The add-in name must be same as the function name
unknownFunctions.Add("D:\AddIn.xlam", "AddInFunction")
'Use the function. The expected result is 30
sheet.Range("A3").Formula = "AddInFunction(10,20)"
Dim fileName As String = "AddIn.xlsx"
workbook.Version = ExcelVersion.Excel2010
workbook.SaveAs(fileName)
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    IAddInFunctions unknownFunctions = workbook.AddInFunctions;
    //Adding the XLAM file reference to AddIn functions
    //NOTE: The add-in name must be same as the function name
    unknownFunctions.Add("AddInFunction");
    //Use the function. The expected result is 30
    sheet.Range["A3"].Formula = "AddInFunction(10,20)";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "AddIn";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);

```

```

IWorksheet sheet = workbook.Worksheets[0];
IAddInFunctions unknownFunctions = workbook.AddInFunctions;
//Adding the XLAM file reference to AddIn functions
//NOTE: The add-in name must be same as the function name
unknownFunctions.Add("AddInFunction");
//Use the function. The expected result is 30
sheet.Range["A3"].Formula = "AddInFunction(10,20)";
//Saving the workbook as stream
FileStream stream = new FileStream("AddIn.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    IAddInFunctions unknownFunctions = workbook.AddInFunctions;
    //Adding the XLAM file reference to AddIn functions
    //NOTE: The add-in name must be same as the function name
    unknownFunctions.Add("AddInFunction");
    //Use the function. The expected result is 30
    sheet.Range["A3"].Formula = "AddInFunction(10,20)";
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("AddIn.
        xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("AddIn.xlsx",
        "application/msexcel", stream);
    }
}

```

Note: If you move the file to another computer, or distribute it, the workbook will expect to find the same Add-In, in the same place, on their computers. But, if the Add-In is moved or deleted from the computer, the workbook won't be able to find it, and your code won't work. Make sure that the Add-In is accessed by locating the .xlam file through the menu (Developer -> Add-ins -> Browse).

Defined Names

Cell ranges can be [defined by names](#) to perform formula calculation. This section explains about creating named ranges and accessing them from workbook or worksheet levels.

The following code shows how to define a named range from workbook level.

C#

```
//Defining a name in workbook level for the cell A1  
IName name = workbook.Names.Add("BookLevelName");  
name.RefersToRange = worksheet.Range["A1"];
```

VB.NET

```
'Defining a name in workbook level for the cell A1  
Dim name As IName = workbook.Names.Add("BookLevelName")  
name.RefersToRange = worksheet.Range("A1")
```

UWP

```
//Defining a name in workbook level for the cell A1  
IName name = workbook.Names.Add("BookLevelName");  
name.RefersToRange = worksheet.Range["A1"];
```

ASP.NET CORE

```
//Defining a name in workbook level for the cell A1  
IName name = workbook.Names.Add("BookLevelName");  
name.RefersToRange = worksheet.Range["A1"];
```

XAMARIN

```
//Defining a name in workbook level for the cell A1  
IName name = workbook.Names.Add("BookLevelName");  
name.RefersToRange = worksheet.Range["A1"];
```

The following code shows how to define a named range from worksheet level.

C#

```
//Defining a name in worksheet level for the cell B1  
IName name = worksheet.Names.Add("SheetLevelName");  
name.RefersToRange = worksheet.Range["B1"];
```

VB.NET

```
'Defining a name in worksheet level for the cell B1  
Dim name As IName = worksheet.Names.Add("SheetLevelName")  
name.RefersToRange = worksheet.Range("B1")
```

UWP

```
//Defining a name in worksheet level for the cell B1  
IName name = worksheet.Names.Add("SheetLevelName");
```

```
name.RefersToRange = worksheet.Range["B1"];
```

ASP.NET CORE

```
//Defining a name in worksheet level for the cell B1
IName name = worksheet.Names.Add("SheetLevelName");
name.RefersToRange = worksheet.Range["B1"];
```

XAMARIN

```
//Defining a name in worksheet level for the cell B1
IName name = worksheet.Names.Add("SheetLevelName");
name.RefersToRange = worksheet.Range["B1"];
```

Named Ranges in Formulas

Following code example illustrates how to create workbook-level named ranges and use it in formulas.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Defining a name in workbook level for the cell A1
    IName name1 = workbook.Names.Add("One");
    name1.RefersToRange = sheet.Range["A1"];
    //Defining a name in workbook level for the cell B1
    IName name2 = workbook.Names.Add("Two");
    name2.RefersToRange = sheet.Range["B1"];
    //Formula using defined names
    sheet.Range["C1"].Formula = "=SUM(One,Two)";
    workbook.SaveAs("Formula.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Defining a name in workbook level for the cell A1
Dim name1 As IName = workbook.Names.Add("One")
name1.RefersToRange = sheet.Range("A1")
'Defining a name in workbook level for the cell B1
Dim name2 As IName = workbook.Names.Add("Two")
name2.RefersToRange = sheet.Range("B1")
'Formula using defined names
sheet.Range("C1").Formula = "=SUM(One,Two)"
workbook.SaveAs("Formula.xlsx")
End Using
```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Defining a name in workbook level for the cell A1
    IName name1 = workbook.Names.Add("One");
    name1.RefersToRange = sheet.Range["A1"];
    //Defining a name in workbook level for the cell B1
    IName name2 = workbook.Names.Add("Two");
    name2.RefersToRange = sheet.Range["B1"];
    //Formula using defined names
    sheet.Range["C1"].Formula = "=SUM(One,Two)";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Formula";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Defining a name in workbook level for the cell A1
    IName name1 = workbook.Names.Add("One");
    name1.RefersToRange = sheet.Range["A1"];
    //Defining a name in workbook level for the cell B1
    IName name2 = workbook.Names.Add("Two");
    name2.RefersToRange = sheet.Range["B1"];
    //Formula using defined names
    sheet.Range["C1"].Formula = "=SUM(One,Two)";
    //Saving the workbook as stream
    FileStream stream = new FileStream("Formula.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;

```

```

application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Defining a name in workbook level for the cell A1
IName name1 = workbook.Names.Add("One");
name1.RefersToRange = sheet.Range["A1"];
//Defining a name in workbook level for the cell B1
IName name2 = workbook.Names.Add("Two");
name2.RefersToRange = sheet.Range["B1"];
//Formula using defined names
sheet.Range["C1"].Formula = "=SUM(One,Two)";
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Formul
a.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Formula.xlsx",
"application/msexcel", stream);
}
}

```

Deleting Named Ranges

Named ranges defined in workbook and worksheet levels can be deleted in different ways. The following code shows the possibilities of deleting named ranges.

C#

```

//Deleting named range object
IName name = workbook.Names[0];
name.Delete();
//Deleting named range from workbook
workbook.Names["BookLevelName"].Delete();
//Deleting named range from worksheet
sheet.Names["SheetLevelName"].Delete();

```

VB.NET

```

'Deleting named range object
Dim name As IName = workbook.Names(0)
name.Delete()
'Deleting named range from workbook
workbook.Names("BookLevelName").Delete()
'Deleting named range from worksheet
sheet.Names("SheetLevelName").Delete()

```


UWP

```
//Deleting named range object
IName name = workbook.Names[0];
name.Delete();
//Deleting named range from workbook
workbook.Names["BookLevelName"].Delete();
//Deleting named range from worksheet
sheet.Names["SheetLevelName"].Delete();
```

ASP.NET CORE

```
//Deleting named range object
IName name = workbook.Names[0];
name.Delete();
//Deleting named range from workbook
workbook.Names["BookLevelName"].Delete();
//Deleting named range from worksheet
sheet.Names["SheetLevelName"].Delete();
```

XAMARIN

```
//Deleting named range object
IName name = workbook.Names[0];
name.Delete();
//Deleting named range from workbook
workbook.Names["BookLevelName"].Delete();
//Deleting named range from worksheet
sheet.Names["SheetLevelName"].Delete();
```

Formula Auditing

Microsoft Excel constantly checks in the background for potential errors in your worksheets, when open. If an error is located (or, at the least, what Excel thinks is an error), then the cell is "flagged" with a small green triangle in the upper-left corner of the cell. Auditing a formula helps to identify the error in it.

In certain cases, these errors can be ignored so that the error will not appear in further error checks. The **IgnoreErrorOptions** property of **IRange** manages different types of errors checks, for example numbers stored as text, formula calculation errors and validation errors.

To know more about IgnoreErrorOptions, please refer ExcelIgnoreError enumeration in API section.

Following code illustrates on how to ignore or set error indicators.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Sets warning if number is entered as text
    sheet.Range["A2:D2"].IgnoreErrorOptions = ExcelIgnoreError.NumberAsText;
    //Ignores all the error warnings
```

```
sheet.Range["A3"].IgnoreErrorOptions = ExcelIgnoreError.None;
workbook.SaveAs("FormulaAuditing.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Sets warning if number is entered as text
sheet.Range("A2:D2").IgnoreErrorOptions = ExcelIgnoreError.NumberAsText
'Ignores all the error warnings
sheet.Range("A3").IgnoreErrorOptions = ExcelIgnoreError.None
workbook.SaveAs("FormulaAuditing.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet sheet = workbook.Worksheets[0];
    //Sets warning if number is entered as text
    sheet.Range["A2:D2"].IgnoreErrorOptions = ExcelIgnoreError.NumberAsText;
    //Ignores all the error warnings
    sheet.Range["A3"].IgnoreErrorOptions = ExcelIgnoreError.None;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "FormulaAuditing";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
```

```

FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet sheet = workbook.Worksheets[0];
//Sets warning if number is entered as text.
sheet.Range["A2:D2"].IgnoreErrorOptions = ExcelIgnoreError.NumberAsText;
//Ignores all the error warnings.
sheet.Range["A3"].IgnoreErrorOptions = ExcelIgnoreError.None;
//Saving the workbook as stream
FileStream file = new FileStream("FormulaAuditing.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(file);
file.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    //Sets warning if number is entered as text
    sheet.Range["A2:D2"].IgnoreErrorOptions = ExcelIgnoreError.NumberAsText;
    //Ignores all the error warnings
    sheet.Range["A3"].IgnoreErrorOptions = ExcelIgnoreError.None;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("FormulaAuditing.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("FormulaAuditing.xlsx", "application/msexcel", stream);
    }
}

```

Calculation Engine

Essential Calculate is now (from v9.1.x.x) integrated with Essential XlsIO, and thus makes it possible to calculate formulas entered at runtime without any additional references or packages.

Note: Do not add reference to Syncfusion.Calculate.Base. It will throw conflict errors as these are already integrated with XlsIO.

Note: Only the formulas that are supported by Calculate engine can be calculated at runtime using Essential XlsIO.

Calculate Options

Calculate engines provides certain options like calculation modes, Recalculate before Save and Enable iterations to perform specific calculation.

Calculation Modes

There are various calculation [modes](#) that enable users to customize formula calculations according to their needs. They are:

- Automatic
- Automatic except for Data Tables
- Manual

Following code illustrates on how to set calculation mode in XlsIO.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Setting calculation mode for a workbook
    workbook.CalculationOptions.CalculationMode = ExcelCalculationMode.Manual;
    workbook.SaveAs("CalculationMode.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Setting calculation mode for a workbook
workbook.CalculationOptions.CalculationMode = ExcelCalculationMode.Manual
workbook.SaveAs("CalculationMode.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
```

```

application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Setting calculation mode for a workbook
workbook.CalculationOptions.CalculationMode = ExcelCalculationMode.Manual;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "CalculationMode";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Setting calculation mode for a workbook
    workbook.CalculationOptions.CalculationMode = ExcelCalculationMode.Manual;
    //Saving the workbook as stream
    FileStream stream = new FileStream("CalculationMode.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Setting calculation mode for a workbook
    workbook.CalculationOptions.CalculationMode = ExcelCalculationMode.Manual;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("CalculationMode.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("CalculationMode.xlsx", "application/msexcel", stream);
}
}
```

Recalculate Before Save

In Manual mode, this option controls whether Microsoft Excel should recalculate the workbook as a part of Save process. You can set this option by using **RecalcOnSave** property of **ICalculationOptions** interface.

C#

```
ICalculationOptions calcOptions = workbook.CalculationOptions;
//Set RecalcOnSave to false to avoid re calculation of workbook while saving
calcOptions.RecalcOnSave = false;
```

VB.NET

```
Dim calcOptions As ICalculationOptions = workbook.CalculationOptions
'Set RecalcOnSave to false to avoid re calculation of workbook while saving
calcOptions.RecalcOnSave = False
```

UWP

```
ICalculationOptions calcOptions = workbook.CalculationOptions;
//Set RecalcOnSave to false to avoid re calculation of workbook while saving
calcOptions.RecalcOnSave = false;
```

ASP.NET CORE

```
ICalculationOptions calcOptions = workbook.CalculationOptions;
//Set RecalcOnSave to false to avoid re calculation of workbook while saving
calcOptions.RecalcOnSave = false;
```

XAMARIN

```
ICalculationOptions calcOptions = workbook.CalculationOptions;
//Set RecalcOnSave to false to avoid re calculation of workbook while saving
calcOptions.RecalcOnSave = false;
```

Iteration

Iteration is the repeated recalculation of a worksheet until a specific numeric condition is met. If a formula refers back to one of its own cells, it must determine how many times the formula should recalculate.

Iteration settings will control the maximum number of iteration and the amount of acceptable change. By default, **IsIterationEnabled** is false, so that Excel does not try to solve accidental circular references.

Following code snippet illustrates how to set the Iterations.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Setting iteration
    workbook.CalculationOptions.IsIterationEnabled = true;
    //Number of times to recalculate
    workbook.CalculationOptions.MaximumIteration = 99;
    //Number of acceptable changes
    workbook.CalculationOptions.MaximumChange = 40;
    workbook.SaveAs("Iteration.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Setting Iteration
workbook.CalculationOptions.IsIterationEnabled = True
'Number of times to recalculate
workbook.CalculationOptions.MaximumIteration = 99
'Number of acceptable changes
workbook.CalculationOptions.MaximumChange = 40
workbook.SaveAs("Iteration.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Setting iteration
    workbook.CalculationOptions.IsIterationEnabled = true;
    //Number of times to recalculate
    workbook.CalculationOptions.MaximumIteration = 99;
    //Number of acceptable changes
    workbook.CalculationOptions.MaximumChange = 40;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Iteration";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
```

```
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Setting iteration
    workbook.CalculationOptions.IsIterationEnabled = true;
    //Number of times to recalculate
    workbook.CalculationOptions.MaximumIteration = 99;
    //Number of acceptable changes
    workbook.CalculationOptions.MaximumChange = 40;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Iteration.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Setting iteration
    workbook.CalculationOptions.IsIterationEnabled = true;
    //Number of times to recalculate
    workbook.CalculationOptions.MaximumIteration = 99;
    //Number of acceptable changes
    workbook.CalculationOptions.MaximumChange = 40;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Iteration.xlsx", "application/msexcel", stream);
    }
    else
    {

```



```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Iteration.xlsx",
"application/msexcel", stream);
}
}
```

Working with Conditional Formatting

Conditional formatting allows to format the contents of a cell dynamically. This can be defined and applied in XlsIO through the **ICConditionalFormat** interface.

Create a Conditional Format

The **ICConditionalFormats** represents a collection of conditional formats for a single **IRange**. One or more conditional formats can be added to the range as follows.

C#

```
//Applying conditional formatting to "A1"
ICConditionalFormats condition = worksheet.Range["A1"].ConditionalFormats;
ICConditionalFormat condition1 = condition.AddCondition();
```

VB.NET

```
'Applying conditional formatting to "A1"
Dim condition As ICConditionalFormats =
worksheet.Range("A1").ConditionalFormats
Dim condition1 As ICConditionalFormat = condition.AddCondition()
```

UWP

```
//Applying conditional formatting to "A1"
ICConditionalFormats condition = worksheet.Range["A1"].ConditionalFormats;
ICConditionalFormat condition1 = condition.AddCondition();
```

ASP.NET CORE

```
//Applying conditional formatting to "A1"
ICConditionalFormats condition = worksheet.Range["A1"].ConditionalFormats;
ICConditionalFormat condition1 = condition.AddCondition();
```

XAMARIN

```
//Applying conditional formatting to "A1"
ICConditionalFormats condition = worksheet.Range["A1"].ConditionalFormats;
ICConditionalFormat condition1 = condition.AddCondition();
```

The target range should meet the criteria, which is set using the **ICConditionalFormat** interface. The desired format type is set through the **ExcelCfType** enumerator, which are the supported conditional format types in XlsIO. Refer to the following code.

C#

```
//Represents conditional format rule that the value in target range should
be between 10 and 20
condition1.FormatType = ExcelCfType.CellValue;
```

```
condition1.Operator = ExcelComparisonOperator.Between;
condition1.FirstFormula = "10";
condition1.SecondFormula = "20";
worksheet.Range["A1"].Text = "Enter a number between 10 and 20";
```

VB.NET

```
'Represents conditional format rule that the value in target range should be
between 10 and 20
condition1.FormatType = ExcelCfType.CellValue
condition1.Operator = ExcelComparisonOperator.Between
condition1.FirstFormula = "10"
condition1.SecondFormula = "20"
worksheet.Range("A1").Text = "Enter a number between 10 and 20"
```

UWP

```
//Represents conditional format rule that the value in target range should
be between 10 and 20
condition1.FormatType = ExcelCfType.CellValue;
condition1.Operator = ExcelComparisonOperator.Between;
condition1.FirstFormula = "10";
condition1.SecondFormula = "20";
worksheet.Range["A1"].Text = "Enter a number between 10 and 20";
```

ASP.NET CORE

```
//Represents conditional format rule that the value in target range should
be between 10 and 20
condition1.FormatType = ExcelCfType.CellValue;
condition1.Operator = ExcelComparisonOperator.Between;
condition1.FirstFormula = "10";
condition1.SecondFormula = "20";
worksheet.Range["A1"].Text = "Enter a number between 10 and 20";
```

XAMARIN

```
//Represents conditional format rule that the value in target range should
be between 10 and 20
condition1.FormatType = ExcelCfType.CellValue;
condition1.Operator = ExcelComparisonOperator.Between;
condition1.FirstFormula = "10";
condition1.SecondFormula = "20";
worksheet.Range["A1"].Text = "Enter a number between 10 and 20";
```

When the criteria set for the target range is satisfied, the defined formats (like the one below) are applied in the order of priority. For more details about conditional format priority, see [Manage conditional formatting rule precedence](#).

C#

```
//Setting format properties to be applied when the above condition is met
condition1.BackColor = ExcelKnownColors.Light_orange;
condition1.IsBold = true;
```

```
condition1.IsItalic = true;
```

VB.NET

```
'Setting format properties to be applied when the above condition is met
condition1.BackColor = ExcelKnownColors.Light_orange
condition1.IsBold = True
condition1.IsItalic = True
```

UWP

```
//Setting format properties to be applied when the above condition is met
condition1.BackColor = ExcelKnownColors.Light_orange;
condition1.IsBold = true;
condition1.IsItalic = true;
```

ASP.NET CORE

```
//Setting format properties to be applied when the above condition is met
condition1.BackColor = ExcelKnownColors.Light_orange;
condition1.IsBold = true;
condition1.IsItalic = true;
```

XAMARIN

```
//Setting format properties to be applied when the above condition is met
condition1.BackColor = ExcelKnownColors.Light_orange;
condition1.IsBold = true;
condition1.IsItalic = true;
```

The following code creates and applies various different conditional formats for different ranges in XlsIO.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "A1"
    IConditionalFormats condition = worksheet.Range["A1"].ConditionalFormats;
    IConditionalFormat condition1 = condition.AddCondition();
    //Represents conditional format rule that the value in target range should be between 10 and 20
    condition1.FormatType = ExcelCfType.CellValue;
    condition1.Operator = ExcelComparisonOperator.Between;
    condition1.FirstFormula = "10";
    condition1.SecondFormula = "20";
    worksheet.Range["A1"].Text = "Enter a number between 10 and 20";
    //Setting back color and font style to be applied for target range
    condition1.BackColor = ExcelKnownColors.Light_orange;
    condition1.IsBold = true;
```

```

condition1.IsItalic = true;
//Applying conditional formatting to "A3"
condition = worksheet.Range["A3"].ConditionalFormats;
IConditionalFormat condition2 = condition.AddCondition();
//Represents conditional format rule that the cell value should be 1000
condition2.FormatType = ExcelCFTType.CellValue;
condition2.Operator = ExcelComparisonOperator.Equal;
condition2.FirstFormula = "1000";
worksheet.Range["A3"].Text = "Enter the Number as 1000";
//Setting fill pattern and back color to target range
condition2.FillPattern = ExcelPattern.LightUpwardDiagonal;
condition2.BackColor = ExcelKnownColors.Yellow;
//Applying conditional formatting to "A5"
condition = worksheet.Range["A5"].ConditionalFormats;
IConditionalFormat condition3 = condition.AddCondition();
//Setting conditional format rule that the cell value for target range
should be less than or equal to 1000
condition3.FormatType = ExcelCFTType.CellValue;
condition3.Operator = ExcelComparisonOperator.LessOrEqual;
condition3.FirstFormula = "1000";
worksheet.Range["A5"].Text = "Enter a Number which is less than or equal to
1000";
//Setting back color to target range
condition3.BackColor = ExcelKnownColors.Light_green;
workbook.SaveAs("ConditionalFormatting.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Applying conditional formatting to "A1"
Dim condition As IConditionalFormats =
worksheet.Range("A1").ConditionalFormats
Dim condition1 As IConditionalFormat = condition.AddCondition()
'Represents conditional format rule that the value in target range should be
between 10 and 20
condition1.FormatType = ExcelCFTType.CellValue
condition1.Operator = ExcelComparisonOperator.Between
condition1.FirstFormula = "10"
condition1.SecondFormula = "20"
worksheet.Range("A1").Text = "Enter a number between 10 and 20"
'Setting back color and font style to be applied for target range
condition1.BackColor = ExcelKnownColors.Light_orange
condition1.IsBold = True
condition1.IsItalic = True
'Applying conditional formatting to "A3"
condition = worksheet.Range("A3").ConditionalFormats
Dim condition2 As IConditionalFormat = condition.AddCondition()
'Represents conditional format rule that the cell value should be 1000
condition2.FormatType = ExcelCFTType.CellValue
condition2.Operator = ExcelComparisonOperator.Equal
condition2.FirstFormula = "1000"

```

```

worksheet.Range("A3").Text = "Enter the Number as 1000"
'Setting fill pattern and back color to target range
condition2.FillPattern = ExcelPattern.LightUpwardDiagonal
condition2.BackColor = ExcelKnownColors.Yellow
'Applying conditional formatting to "A5"
condition = worksheet.Range("A5").ConditionalFormats
Dim condition3 As IConditionalFormat = condition.AddCondition()
'Setting conditional format rule that the cell value for target range should
be less than or equal to 1000
condition3.FormatType = ExcelCFTType.CellValue
condition3.Operator = ExcelComparisonOperator.LessOrEqual
condition3.FirstFormula = "1000"
worksheet.Range("A5").Text = "Enter a Number which is less than or equal to
1000"
'Setting back color to target range
condition3.BackColor = ExcelKnownColors.Light_green
workbook.SaveAs ("ConditionalFormatting.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "A1"
    IConditionalFormats condition = worksheet.Range["A1"].ConditionalFormats;
    IConditionalFormat condition1 = condition.AddCondition();
    //Represents conditional format rule that the value in target range should
    be between 10 and 20
    condition1.FormatType = ExcelCFTType.CellValue;
    condition1.Operator = ExcelComparisonOperator.Between;
    condition1.FirstFormula = "10";
    condition1.SecondFormula = "20";
    worksheet.Range["A1"].Text = "Enter a number between 10 and 20";
    //Setting back color and font style to be applied for target range
    condition1.BackColor = ExcelKnownColors.Light_orange;
    condition1.IsBold = true;
    condition1.IsItalic = true;
    //Applying conditional formatting to "A3"
    condition = worksheet.Range["A3"].ConditionalFormats;
    IConditionalFormat condition2 = condition.AddCondition();
    //Represents conditional format rule that the cell value should be 1000
    condition2.FormatType = ExcelCFTType.CellValue;
    condition2.Operator = ExcelComparisonOperator.Equal;
    condition2.FirstFormula = "1000";
    worksheet.Range["A3"].Text = "Enter the Number as 1000";
    //Setting fill pattern and back color to target range
    condition2.FillPattern = ExcelPattern.LightUpwardDiagonal;
    condition2.BackColor = ExcelKnownColors.Yellow;
    //Applying conditional formatting to "A5"
    condition = worksheet.Range["A5"].ConditionalFormats;
    IConditionalFormat condition3 = condition.AddCondition();
}

```

```

//Setting conditional format rule that the cell value for target range
should be less than or equal to 1000
condition3.FormatType = ExcelCfType.CellValue;
condition3.Operator = ExcelComparisonOperator.LessOrEqual;
condition3.FirstFormula = "1000";
worksheet.Range["A5"].Text = "Enter a Number which is less than or equal to
1000";
//Setting back color to target range
condition3.BackColor = ExcelKnownColors.Light_green;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ConditionalFormatting";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "A1"
    IConditionalFormats condition = worksheet.Range["A1"].ConditionalFormats;
    IConditionalFormat condition1 = condition.AddCondition();
    //Represents conditional format rule that the value in target range should
be between 10 and 20
    condition1.FormatType = ExcelCfType.CellValue;
    condition1.Operator = ExcelComparisonOperator.Between;
    condition1.FirstFormula = "10";
    condition1.SecondFormula = "20";
    worksheet.Range["A1"].Text = "Enter a number between 10 and 20";
    //Setting back color and font style to be applied for target range
    condition1.BackColor = ExcelKnownColors.Light_orange;
    condition1.IsBold = true;
    condition1.IsItalic = true;
    //Applying conditional formatting to "A3"
    condition = worksheet.Range["A3"].ConditionalFormats;
    IConditionalFormat condition2 = condition.AddCondition();
    //Represents conditional format rule that the cell value should be 1000
    condition2.FormatType = ExcelCfType.CellValue;
    condition2.Operator = ExcelComparisonOperator.Equal;
    condition2.FirstFormula = "1000";
    worksheet.Range["A3"].Text = "Enter the Number as 1000";
    //Setting fill pattern and back color to target range
    condition2.FillPattern = ExcelPattern.LightUpwardDiagonal;
    condition2.BackColor = ExcelKnownColors.Yellow;
    //Applying conditional formatting to "A5"
    condition = worksheet.Range["A5"].ConditionalFormats;

```

```

IConditionalFormat condition3 = condition.AddCondition();
//Setting conditional format rule that the cell value for target range
should be less than or equal to 1000
condition3.FormatType = ExcelCFTType.CellValue;
condition3.Operator = ExcelComparisonOperator.LessOrEqual;
condition3.FirstFormula = "1000";
worksheet.Range["A5"].Text = "Enter a Number which is less than or equal to
1000";
//Setting back color to target range
condition3.BackColor = ExcelKnownColors.Light_green;
//Saving the workbook as stream
FileStream stream = new FileStream("ConditionalFormatting.xlsx",
FileMode.Create, FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Applying conditional formatting to "A1"
IConditionalFormats condition = worksheet.Range["A1"].ConditionalFormats;
IConditionalFormat condition1 = condition.AddCondition();
//Represents conditional format rule that the value in target range should
be between 10 and 20
condition1.FormatType = ExcelCFTType.CellValue;
condition1.Operator = ExcelComparisonOperator.Between;
condition1.FirstFormula = "10";
condition1.SecondFormula = "20";
worksheet.Range["A1"].Text = "Enter a number between 10 and 20";
//Setting back color and font style to be applied for target range
condition1.BackColor = ExcelKnownColors.Light_orange;
condition1.IsBold = true;
condition1.IsItalic = true;
//Applying conditional formatting to "A3"
condition = worksheet.Range["A3"].ConditionalFormats;
IConditionalFormat condition2 = condition.AddCondition();
//Represents conditional format rule that the cell value should be 1000
condition2.FormatType = ExcelCFTType.CellValue;
condition2.Operator = ExcelComparisonOperator.Equal;
condition2.FirstFormula = "1000";
worksheet.Range["A3"].Text = "Enter the Number as 1000";
//Setting fill pattern and back color to target range
condition2.FillPattern = ExcelPattern.LightUpwardDiagonal;
condition2.BackColor = ExcelKnownColors.Yellow;
//Applying conditional formatting to "A5"
condition = worksheet.Range["A5"].ConditionalFormats;
IConditionalFormat condition3 = condition.AddCondition();
//Setting conditional format rule that the cell value for target range
should be less than or equal to 1000
condition3.FormatType = ExcelCFTType.CellValue;

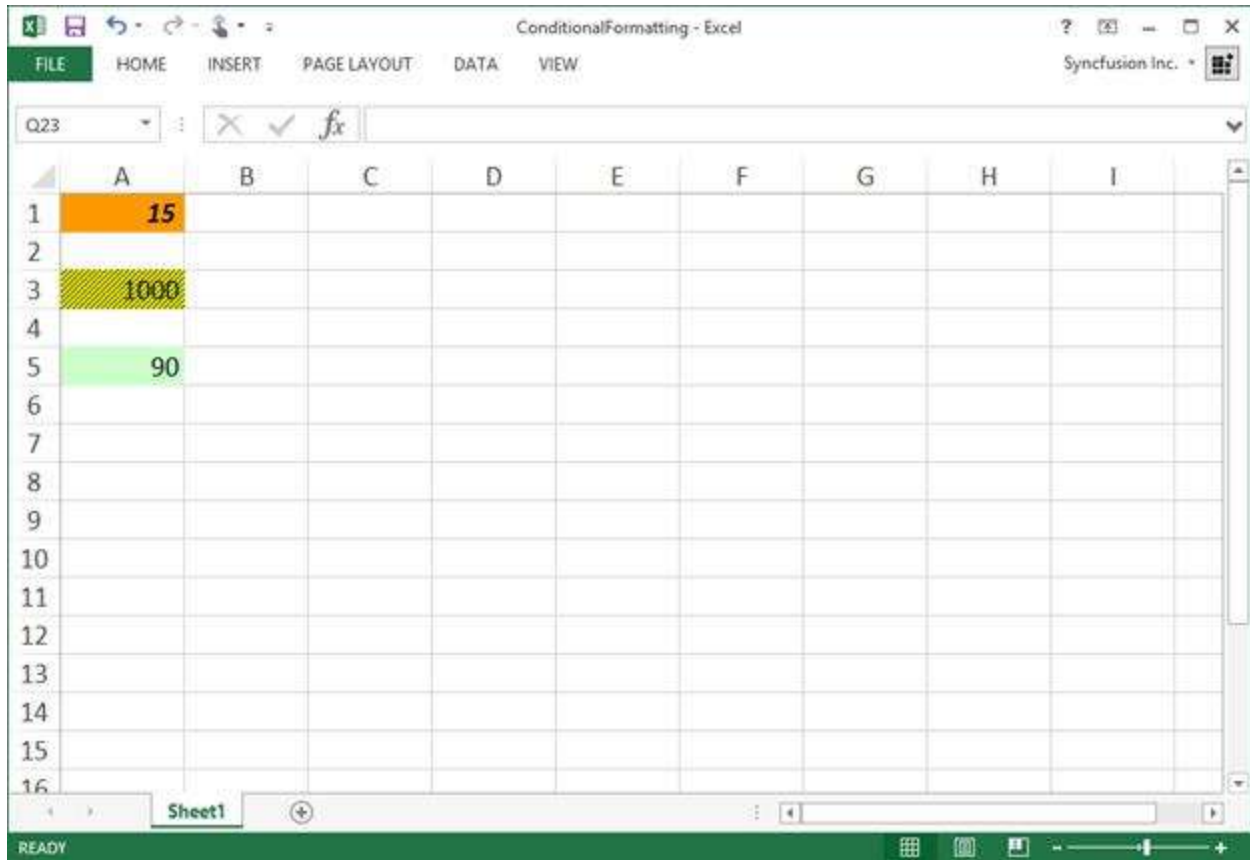
```

```
condition3.Operator = ExcelComparisonOperator.LessOrEqual;
condition3.FirstFormula = "1000";
worksheet.Range["A5"].Text = "Enter a Number which is less than or equal to 1000";
//Setting back color to target range
condition3.BackColor = ExcelKnownColors.Light_green;
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android, and iOS platforms. Refer to the xlsio/xamarin section for
//respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("ConditionalFormatting.xlsx", "application/msexcel", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ConditionalFormatting.xlsx", "application/msexcel", stream);
}
}
```

Note: Excel allows the addition of a maximum of three conditions for the same cell in the Biff8 format and XlsIO. However, this restriction is removed from the Excel 2007 formats.

Note: The conditional formats for a single range should be added in descending order in XlsIO.

When proper criteria is met, the output file looks as follows:



Reading Conditional Formats in XlsIO

XlsIO also reads conditional formats from existing excel workbook. The following code example illustrates this.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Read conditional formatting settings
    string formatType =
        worksheet.Range["A1"].ConditionalFormats[0].FormatType.ToString();
    string cfOperator =
        worksheet.Range["A1"].ConditionalFormats[0].Operator.ToString();
    string backColor =
        worksheet.Range["A1"].ConditionalFormats[0].BackColor.ToString();
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
```

```

application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Read conditional formatting settings
Dim formatType As String =
worksheet.Range("A1").ConditionalFormats(0).FormatType.ToString()
Dim cfOperator As String =
worksheet.Range("A1").ConditionalFormats(0).Operator.ToString()
Dim backColor As String =
worksheet.Range("A1").ConditionalFormats(0).BackColor.ToString()
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Read conditional formatting settings
    string formatType =
worksheet.Range["A1"].ConditionalFormats[0].FormatType.ToString();
    string cfOperator =
worksheet.Range["A1"].ConditionalFormats[0].Operator.ToString();
    string backColor =
worksheet.Range["A1"].ConditionalFormats[0].BackColor.ToString();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;

```

```

FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream,
    ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Read conditional formatting settings
string formatType =
    worksheet.Range["A1"].ConditionalFormats[0].FormatType.ToString();
string cfOperator =
    worksheet.Range["A1"].ConditionalFormats[0].Operator.ToString();
string backColor =
    worksheet.Range["A1"].ConditionalFormats[0].BackColor.ToString();
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    /"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Read conditional formatting settings
    string formatType =
        worksheet.Range["A1"].ConditionalFormats[0].FormatType.ToString();
    string cfOperator =
        worksheet.Range["A1"].ConditionalFormats[0].Operator.ToString();
    string backColor =
        worksheet.Range["A1"].ConditionalFormats[0].BackColor.ToString();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output.xlsx", "application/msexcel", stream);
    }
    else
    {

```

```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}
```

Removing Conditional Formats

All the conditional formats for a specified range can be removed using the **Remove** method. This is illustrated as follows.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Removing conditional format for a specified range
    worksheet.Range["E5"].ConditionalFormats.Remove();
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Removing conditional format for a specified range
worksheet.Range("E5").ConditionalFormats.Remove()
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Removing conditional format for a specified range
    worksheet.Range["E5"].ConditionalFormats.Remove();
    //Initializes FileSavePicker
}
```

```

FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream,
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Removing conditional format for a specified range
    worksheet.Range["E5"].ConditionalFormats.Remove();
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Removing conditional format for a specified range
    worksheet.Range["E5"].ConditionalFormats.Remove();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
}

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Removing Conditional Formats at specified index value

A particular conditional format at the specified range can be removed by using the *RemoveAt** method as follows.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Removing first conditional Format at the specified Range
    worksheet.Range["E5"].ConditionalFormats.RemoveAt(0);
    workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Removing first conditional Format at the specified Range
worksheet.Range("E5").ConditionalFormats.RemoveAt(0)
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
}

```

```

StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Removing first conditional Format at the specified Range
worksheet.Range["E5"].ConditionalFormats.RemoveAt(0);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Removing first conditional Format at the specified Range
worksheet.Range["E5"].ConditionalFormats.RemoveAt(0);
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
ple.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Removing first conditional Format at the specified Range
worksheet.Range["E5"].ConditionalFormats.RemoveAt(0);
//Saving the workbook as stream

```

```

MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android, and iOS platforms. Refer to the xlsio/xamarin section for
//respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Removing Conditional Formats from entire sheet

The entire conditional formats from the worksheet can be removed as follows.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Removing Conditional Formatting Settings From Entire Sheet
worksheet.UsedRange.Clear(ExcelClearOptions.ClearConditionalFormats);
workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Removing Conditional Formatting Settings From Entire Sheet
worksheet.UsedRange.Clear(ExcelClearOptions.ClearConditionalFormats)
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;

```



```

application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Removing Conditional Formatting Settings From Entire Sheet
worksheet.UsedRange.Clear(ExcelClearOptions.ClearConditionalFormats);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Removing Conditional Formatting Settings From Entire Sheet
worksheet.UsedRange.Clear(ExcelClearOptions.ClearConditionalFormats);
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
ple.xlsx");
}

```

```

IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Removing Conditional Formatting Settings From Entire Sheet
worksheet.UsedRange.Clear(ExcelClearOptions.ClearConditionalFormats);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Using FormulaR1C1 property in Conditional Formats

XlsIO sets the formula for the conditional format in R1C1-style notation. The following code example illustrates this.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Using FormulaR1C1 property in Conditional Formatting
IConditionalFormats condition =
worksheet.Range["E5:E18"].ConditionalFormats;
IConditionalFormat condition1 = condition.AddCondition();
condition1.FirstFormulaR1C1 = "=R[1]C[0]";
condition1.SecondFormulaR1C1 = "=R[1]C[1]";
workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Using FormulaR1C1 property in Conditional Formatting

```

```

Dim condition As IConditionalFormats =
worksheet.Range("E5:E18").ConditionalFormats
Dim condition1 As IConditionalFormat = condition.AddCondition()
condition1.FirstFormulaR1C1 = "=R[1]C[0]"
condition1.SecondFormulaR1C1 = "=R[1]C[1]"
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Using FormulaR1C1 property in Conditional Formatting
    IConditionalFormats condition =
worksheet.Range["E5:E18"].ConditionalFormats;
    IConditionalFormat condition1 = condition.AddCondition();
    condition1.FirstFormulaR1C1 = "=R[1]C[0]";
    condition1.SecondFormulaR1C1 = "=R[1]C[1]";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Using FormulaR1C1 property in Conditional Formatting
    IConditionalFormats condition =
worksheet.Range["E5:E18"].ConditionalFormats;
    IConditionalFormat condition1 = condition.AddCondition();
    condition1.FirstFormulaR1C1 = "=R[1]C[0]";
    condition1.SecondFormulaR1C1 = "=R[1]C[1]";
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Using FormulaR1C1 property in Conditional Formatting
    IConditionalFormats condition =
        worksheet.Range["E5:E18"].ConditionalFormats;
    IConditionalFormat condition1 = condition.AddCondition();
    condition1.FirstFormulaR1C1 = "=R[1]C[0]";
    condition1.SecondFormulaR1C1 = "=R[1]C[1]";
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
        "application/msexcel", stream);
    }
}

```

Format Unique and Duplicate Values

Format unique and duplicate values of an Excel range using conditional formatting. The values, Unique and Duplicate of the enumeration ExcelCfType helps to achieve the requirement.

The below code example shows how to format unique and duplicate values using conditional formatting in XlsIO.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Fill worksheet with data
    worksheet.Range["A1:B1"].Merge();
    worksheet.Range["A1:B1"].CellStyle.Font.RGBColor = Color.FromArgb(255, 102,
    102, 255);
    worksheet.Range["A1:B1"].CellStyle.Font.Size = 14;
}

```

```

worksheet.Range["A1:B1"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignCenter;
worksheet.Range["A1"].Text = "Global Internet Usage";
worksheet.Range["A1:B1"].CellStyle.Font.Bold = true;
worksheet.Range["A3:B21"].CellStyle.Font.RGBColor = Color.FromArgb(255, 64,
64, 64);
worksheet.Range["A3:B3"].CellStyle.Font.Bold = true;
worksheet.Range["B3"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignRight;
worksheet.Range["A3"].Text = "Country";
worksheet.Range["A4"].Text = "Northern America";
worksheet.Range["A5"].Text = "Central America";
worksheet.Range["A6"].Text = "The Caribbean";
worksheet.Range["A7"].Text = "South America";
worksheet.Range["A8"].Text = "Northern Europe";
worksheet.Range["A9"].Text = "Eastern Europe";
worksheet.Range["A10"].Text = "Western Europe";
worksheet.Range["A11"].Text = "Southern Europe";
worksheet.Range["A12"].Text = "Northern Africa";
worksheet.Range["A13"].Text = "Eastern Africa";
worksheet.Range["A14"].Text = "Middle Africa";
worksheet.Range["A15"].Text = "Western Africa";
worksheet.Range["A16"].Text = "Southern Africa";
worksheet.Range["A17"].Text = "Central Asia";
worksheet.Range["A18"].Text = "Eastern Asia";
worksheet.Range["A19"].Text = "Southern Asia";
worksheet.Range["A20"].Text = "SouthEast Asia";
worksheet.Range["A21"].Text = "Oceania";
worksheet.Range["B3"].Text = "Usage";
worksheet.Range["B4"].Value = "88%";
worksheet.Range["B5"].Value = "61%";
worksheet.Range["B6"].Value = "49%";
worksheet.Range["B7"].Value = "68%";
worksheet.Range["B8"].Value = "94%";
worksheet.Range["B9"].Value = "74%";
worksheet.Range["B10"].Value = "90%";
worksheet.Range["B11"].Value = "77%";
worksheet.Range["B12"].Value = "49%";
worksheet.Range["B13"].Value = "27%";
worksheet.Range["B14"].Value = "12%";
worksheet.Range["B15"].Value = "39%";
worksheet.Range["B16"].Value = "51%";
worksheet.Range["B17"].Value = "50%";
worksheet.Range["B18"].Value = "58%";
worksheet.Range["B19"].Value = "36%";
worksheet.Range["B20"].Value = "58%";
worksheet.Range["B21"].Value = "69%";
worksheet.SetColumnWidth(1, 23.45);
worksheet.SetColumnWidth(2, 8.09);
IConditionalFormats conditionalFormats =
worksheet.Range["A4:B21"].ConditionalFormats;
IConditionalFormat condition = conditionalFormats.AddCondition();
//conditional format to set duplicate format type
condition.FormatType = ExcelCfType.Duplicate;
condition.BackColorRGB = Color.FromArgb(255, 255, 199, 206);
//Saves the Excel
workbook.SaveAs("Output.xlsx");

```

```
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Fill worksheet with data
worksheet.Range("A1:B1").Merge()
worksheet.Range("A1:B1").CellStyle.Font.RGBColor = Color.FromArgb(255, 102,
102, 255)
worksheet.Range("A1:B1").CellStyle.Font.Size = 14
worksheet.Range("A1:B1").CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignCenter
worksheet.Range("A1").Text = "Global Internet Usage"
worksheet.Range("A1:B1").CellStyle.Font.Bold = True
worksheet.Range("A3:B21").CellStyle.Font.RGBColor = Color.FromArgb(255, 64,
64, 64)
worksheet.Range("A3:B3").CellStyle.Font.Bold = True
worksheet.Range("B3").CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignRight
worksheet.Range("A3").Text = "Country"
worksheet.Range("A4").Text = "Northern America"
worksheet.Range("A5").Text = "Central America"
worksheet.Range("A6").Text = "The Caribbean"
worksheet.Range("A7").Text = "South America"
worksheet.Range("A8").Text = "Northern Europe"
worksheet.Range("A9").Text = "Eastern Europe"
worksheet.Range("A10").Text = "Western Europe"
worksheet.Range("A11").Text = "Southern Europe"
worksheet.Range("A12").Text = "Northern Africa"
worksheet.Range("A13").Text = "Eastern Africa"
worksheet.Range("A14").Text = "Middle Africa"
worksheet.Range("A15").Text = "Western Africa"
worksheet.Range("A16").Text = "Southern Africa"
worksheet.Range("A17").Text = "Central Asia"
worksheet.Range("A18").Text = "Eastern Asia"
worksheet.Range("A19").Text = "Southern Asia"
worksheet.Range("A20").Text = "SouthEast Asia"
worksheet.Range("A21").Text = "Oceania"
worksheet.Range("B3").Text = "Usage"
worksheet.Range("B4").Value = "88%"
worksheet.Range("B5").Value = "61%"
worksheet.Range("B6").Value = "49%"
worksheet.Range("B7").Value = "68%"
worksheet.Range("B8").Value = "94%"
worksheet.Range("B9").Value = "74%"
worksheet.Range("B10").Value = "90%"
worksheet.Range("B11").Value = "77%"
worksheet.Range("B12").Value = "49%"
worksheet.Range("B13").Value = "27%"
worksheet.Range("B14").Value = "12%"
worksheet.Range("B15").Value = "39%"
worksheet.Range("B16").Value = "51%"

```

```

worksheet.Range("B17").Value = "50%"
worksheet.Range("B18").Value = "58%"
worksheet.Range("B19").Value = "36%"
worksheet.Range("B20").Value = "58%"
worksheet.Range("B21").Value = "69%"
worksheet.SetColumnWidth(1, 23.45)
worksheet.SetColumnWidth(2, 8.09)
'conditional format to set duplicate format type
Dim conditionalFormats As IConditionalFormats =
worksheet.Range("A4:B21").ConditionalFormats
Dim condition As IConditionalFormat = conditionalFormats.AddCondition()
condition.FormatType = ExcelCFTType.Duplicate
condition.BackColorRGB = Color.FromArgb(255, 255, 199, 206)
'Saves the Excel
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Fill worksheet with data
    worksheet.Range["A1:B1"].Merge();
    worksheet.Range["A1:B1"].CellStyle.Font.RGBColor = Color.FromArgb(255, 102,
    102, 255);
    worksheet.Range["A1:B1"].CellStyle.Font.Size = 14;
    worksheet.Range["A1:B1"].CellStyle.HorizontalAlignment =
    ExcelHAlign.HAlignCenter;
    worksheet.Range["A1"].Text = "Global Internet Usage";
    worksheet.Range["A1:B1"].CellStyle.Font.Bold = true;
    worksheet.Range["A3:B21"].CellStyle.Font.RGBColor = Color.FromArgb(255, 64,
    64, 64);
    worksheet.Range["A3:B3"].CellStyle.Font.Bold = true;
    worksheet.Range["B3"].CellStyle.HorizontalAlignment =
    ExcelHAlign.HAlignRight;
    worksheet.Range["A3"].Text = "Country";
    worksheet.Range["A4"].Text = "Northern America";
    worksheet.Range["A5"].Text = "Central America";
    worksheet.Range["A6"].Text = "The Caribbean";
    worksheet.Range["A7"].Text = "South America";
    worksheet.Range["A8"].Text = "Northern Europe";
    worksheet.Range["A9"].Text = "Eastern Europe";
    worksheet.Range["A10"].Text = "Western Europe";
    worksheet.Range["A11"].Text = "Southern Europe";
    worksheet.Range["A12"].Text = "Northern Africa";
    worksheet.Range["A13"].Text = "Eastern Africa";
    worksheet.Range["A14"].Text = "Middle Africa";
    worksheet.Range["A15"].Text = "Western Africa";
    worksheet.Range["A16"].Text = "Southern Africa";
    worksheet.Range["A17"].Text = "Central Asia";
    worksheet.Range["A18"].Text = "Eastern Asia";
    worksheet.Range["A19"].Text = "Southern Asia";
}

```

```

worksheet.Range["A20"].Text = "SouthEast Asia";
worksheet.Range["A21"].Text = "Oceania";
worksheet.Range["B3"].Text = "Usage";
worksheet.Range["B4"].Value = "88%";
worksheet.Range["B5"].Value = "61%";
worksheet.Range["B6"].Value = "49%";
worksheet.Range["B7"].Value = "68%";
worksheet.Range["B8"].Value = "94%";
worksheet.Range["B9"].Value = "74%";
worksheet.Range["B10"].Value = "90%";
worksheet.Range["B11"].Value = "77%";
worksheet.Range["B12"].Value = "49%";
worksheet.Range["B13"].Value = "27%";
worksheet.Range["B14"].Value = "12%";
worksheet.Range["B15"].Value = "39%";
worksheet.Range["B16"].Value = "51%";
worksheet.Range["B17"].Value = "50%";
worksheet.Range["B18"].Value = "58%";
worksheet.Range["B19"].Value = "36%";
worksheet.Range["B20"].Value = "58%";
worksheet.Range["B21"].Value = "69%";
worksheet.SetColumnWidth(1, 23.45);
worksheet.SetColumnWidth(2, 8.09);
IConditionalFormats conditionalFormats =
worksheet.Range["A4:B21"].ConditionalFormats;
IConditionalFormat condition = conditionalFormats.AddCondition();
//conditional format to set duplicate format type
condition.FormatType = ExcelCfType.Duplicate;
condition.BackColorRGB = Color.FromArgb(255, 255, 199, 206);
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile outputStorageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(outputStorageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Fill worksheet with data
    worksheet.Range["A1:B1"].Merge();
    worksheet.Range["A1:B1"].CellStyle.Font.RGBColor = Color.FromArgb(255, 102,
    102, 255);
    worksheet.Range["A1:B1"].CellStyle.Font.Size = 14;
    worksheet.Range["A1:B1"].CellStyle.HorizontalAlignment =
    ExcelHAlign.HAlignCenter;
    worksheet.Range["A1"].Text = "Global Internet Usage";
}

```



```

worksheet.Range["A1:B1"].CellStyle.Font.Bold = true;
worksheet.Range["A3:B21"].CellStyle.Font.RGBColor = Color.FromArgb(255, 64,
64, 64);
worksheet.Range["A3:B3"].CellStyle.Font.Bold = true;
worksheet.Range["B3"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignRight;
worksheet.Range["A3"].Text = "Country";
worksheet.Range["A4"].Text = "Northern America";
worksheet.Range["A5"].Text = "Central America";
worksheet.Range["A6"].Text = "The Caribbean";
worksheet.Range["A7"].Text = "South America";
worksheet.Range["A8"].Text = "Northern Europe";
worksheet.Range["A9"].Text = "Eastern Europe";
worksheet.Range["A10"].Text = "Western Europe";
worksheet.Range["A11"].Text = "Southern Europe";
worksheet.Range["A12"].Text = "Northern Africa";
worksheet.Range["A13"].Text = "Eastern Africa";
worksheet.Range["A14"].Text = "Middle Africa";
worksheet.Range["A15"].Text = "Western Africa";
worksheet.Range["A16"].Text = "Southern Africa";
worksheet.Range["A17"].Text = "Central Asia";
worksheet.Range["A18"].Text = "Eastern Asia";
worksheet.Range["A19"].Text = "Southern Asia";
worksheet.Range["A20"].Text = "SouthEast Asia";
worksheet.Range["A21"].Text = "Oceania";
worksheet.Range["B3"].Text = "Usage";
worksheet.Range["B4"].Value = "88%";
worksheet.Range["B5"].Value = "61%";
worksheet.Range["B6"].Value = "49%";
worksheet.Range["B7"].Value = "68%";
worksheet.Range["B8"].Value = "94%";
worksheet.Range["B9"].Value = "74%";
worksheet.Range["B10"].Value = "90%";
worksheet.Range["B11"].Value = "77%";
worksheet.Range["B12"].Value = "49%";
worksheet.Range["B13"].Value = "27%";
worksheet.Range["B14"].Value = "12%";
worksheet.Range["B15"].Value = "39%";
worksheet.Range["B16"].Value = "51%";
worksheet.Range["B17"].Value = "50%";
worksheet.Range["B18"].Value = "58%";
worksheet.Range["B19"].Value = "36%";
worksheet.Range["B20"].Value = "58%";
worksheet.Range["B21"].Value = "69%";
worksheet.SetColumnWidth(1, 23.45);
worksheet.SetColumnWidth(2, 8.09);
IConditionalFormats conditionalFormats =
worksheet.Range["A4:B21"].ConditionalFormats;
IConditionalFormat condition = conditionalFormats.AddCondition();
//conditional format to set duplicate format type
condition.FormatType = ExcelCfType.Duplicate;
condition.BackColorRGB = Color.FromArgb(255, 255, 199, 206);
//Saves the excel document to MemoryStream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();

```

```
}

```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Fill worksheet with data
    worksheet.Range["A1:B1"].Merge();
    worksheet.Range["A1:B1"].CellStyle.Font.RGBColor = Color.FromArgb(255, 102,
    102, 255);
    worksheet.Range["A1:B1"].CellStyle.Font.Size = 14;
    worksheet.Range["A1:B1"].CellStyle.HorizontalAlignment =
    ExcelHAlign.HAlignCenter;
    worksheet.Range["A1"].Text = "Global Internet Usage";
    worksheet.Range["A1:B1"].CellStyle.Font.Bold = true;
    worksheet.Range["A3:B21"].CellStyle.Font.RGBColor = Color.FromArgb(255, 64,
    64, 64);
    worksheet.Range["A3:B3"].CellStyle.Font.Bold = true;
    worksheet.Range["B3"].CellStyle.HorizontalAlignment =
    ExcelHAlign.HAlignRight;
    worksheet.Range["A3"].Text = "Country";
    worksheet.Range["A4"].Text = "Northern America";
    worksheet.Range["A5"].Text = "Central America";
    worksheet.Range["A6"].Text = "The Caribbean";
    worksheet.Range["A7"].Text = "South America";
    worksheet.Range["A8"].Text = "Northern Europe";
    worksheet.Range["A9"].Text = "Eastern Europe";
    worksheet.Range["A10"].Text = "Western Europe";
    worksheet.Range["A11"].Text = "Southern Europe";
    worksheet.Range["A12"].Text = "Northern Africa";
    worksheet.Range["A13"].Text = "Eastern Africa";
    worksheet.Range["A14"].Text = "Middle Africa";
    worksheet.Range["A15"].Text = "Western Africa";
    worksheet.Range["A16"].Text = "Southern Africa";
    worksheet.Range["A17"].Text = "Central Asia";
    worksheet.Range["A18"].Text = "Eastern Asia";
    worksheet.Range["A19"].Text = "Southern Asia";
    worksheet.Range["A20"].Text = "SouthEast Asia";
    worksheet.Range["A21"].Text = "Oceania";
    worksheet.Range["B3"].Text = "Usage";
    worksheet.Range["B4"].Value = "88%";
    worksheet.Range["B5"].Value = "61%";
    worksheet.Range["B6"].Value = "49%";
    worksheet.Range["B7"].Value = "68%";
    worksheet.Range["B8"].Value = "94%";
    worksheet.Range["B9"].Value = "74%";
    worksheet.Range["B10"].Value = "90%";
    worksheet.Range["B11"].Value = "77%";
    worksheet.Range["B12"].Value = "49%";
    worksheet.Range["B13"].Value = "27%";
    worksheet.Range["B14"].Value = "12%";
    worksheet.Range["B15"].Value = "39%";
}
```

```

worksheet.Range["B16"].Value = "51%";
worksheet.Range["B17"].Value = "50%";
worksheet.Range["B18"].Value = "58%";
worksheet.Range["B19"].Value = "36%";
worksheet.Range["B20"].Value = "58%";
worksheet.Range["B21"].Value = "69%";
worksheet.SetColumnWidth(1, 23.45);
worksheet.SetColumnWidth(2, 8.09);
IConditionalFormats conditionalFormats =
worksheet.Range["A4:B21"].ConditionalFormats;
IConditionalFormat condition = conditionalFormats.AddCondition();
//conditional format to set duplicate format type
condition.FormatType = ExcelCFTType.Duplicate;
condition.BackColorRGB = Color.FromArgb(255, 255, 199, 206);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

The following screenshot represents generated Excel file with unique and duplicate conditional format in XlsIO.

	A	B
1	GLOBAL INTERNET USAGE	
2		
3	Country	Usage
4	Northern America	88%
5	Central America	61%
6	The Caribbean	49%
7	South America	68%
8	Northern Europe	94%
9	Eastern Europe	74%
10	Western Europe	90%
11	Southern Europe	77%
12	Northern Africa	49%
13	Eastern Africa	27%
14	Middle Africa	12%
15	Western Africa	39%
16	Southern Africa	51%
17	Central Asia	50%
18	Eastern Asia	58%
19	Southern Asia	36%
20	SouthEast Asia	58%
21	Oceania	69%

Format Top or Bottom Values

Top/Bottom rule in conditional formatting is used to highlight the top or bottom ranked cells in a data range. Top/Bottom conditional formatting rule can be created and customized using the **ITopBottom** interface in XlsIO.

The properties of **ITopBottom** interface are:

- **Type** - Specifies whether the rank is evaluated from the top or bottom.
- **Percent** - Specifies whether the rank is determined by a percentage value.
- **Rank** - Specifies the maximum number or percentage of cells to be highlighted.

The following screenshot represents the input template of conditional formatting.

Students	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Subject 10	Total	Average	Rank
Student 1	99	97	93	100	91	92	92	95	91	89	939	94	3
Student 2	86	88	90	92	91	87	96	75	80	92	877	88	9
Student 3	91	92	97	94	91	92	95	99	100	94	945	95	1
Student 4	64	71	82	80	63	71	88	78	76	83	756	76	26
Student 5	91	85	79	92	86	81	83	90	82	82	851	85	17
Student 6	90	81	90	83	82	92	95	89	87	88	877	88	9
Student 7	82	89	94	91	86	87	80	83	86	80	858	86	16
Student 8	77	78	60	79	65	77	80	73	70	81	740	74	29
Student 9	71	82	69	75	69	81	70	72	74	84	747	75	27
Student 10	85	81	84	88	83	81	89	88	82	85	846	85	18
Student 11	85	75	79	78	82	86	81	70	79	86	801	80	25
Student 12	91	92	90	100	91	91	92	92	92	91	922	92	4
Student 13	94	81	93	69	82	90	91	92	89	81	862	86	14
Student 14	87	89	87	86	82	80	84	92	90	89	866	87	12
Student 15	78	75	71	75	79	70	72	73	76	77	746	75	28
Student 16	93	91	90	89	82	85	87	88	87	84	876	88	11
Student 17	88	82	80	81	84	81	80	82	91	87	836	84	20
Student 18	93	94	95	92	90	90	98	100	99	94	945	95	1
Student 19	84	90	91	93	90	92	91	81	85	84	881	88	7
Student 20	83	75	74	76	78	89	90	87	82	88	822	82	22
Student 21	89	87	83	80	91	72	92	85	89	94	862	86	14
Student 22	89	87	91	90	90	83	87	91	90	94	892	89	5
Student 23	87	91	90	90	81	79	78	79	80	87	842	84	19
Student 24	79	86	75	75	71	82	85	80	93	94	820	82	23
Student 25	80	79	87	89	91	90	94	93	91	94	888	89	6
Student 26	69	67	71	69	71	69	69	60	71	73	689	69	30
Student 27	73	77	78	82	84	90	91	75	80	85	815	82	24
Student 28	93	87	85	83	71	80	82	85	81	84	831	83	21
Student 29	85	90	93	91	91	92	87	85	85	80	879	88	8
Student 30	89	87	83	86	81	80	92	80	94	92	864	86	13

Top/Bottom 'n' rank values

The below code example shows how to format top 10 rank values from the given data range using `ITopBottom` Type and `Rank` properties in XlsIO.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("CFTemplate.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "N6:N35".
    IConditionalFormats formats = worksheet.Range["N6:N35"].ConditionalFormats;
    IConditionalFormat format = formats.AddCondition();
    //Applying top or bottom rule in the conditional formatting.
    format.FormatType = ExcelCFTType.TopBottom;
    ITopBottom topBottom = format.TopBottom;
    //Set type as Top for TopBottom rule.
    topBottom.Type = ExcelCFTTopBottomType.Top;
    //Set rank value for the TopBottom rule.
    topBottom.Rank = 10;
    //Set color for Conditional Formatting.
}
```

```
format.BackColorRGB = System.Drawing.Color.FromArgb(51, 153, 102);
//Saves the Excel
workbook.SaveAs("TopBottom.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("CFTemplate.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Applying conditional formatting to "N6:N35".
Dim formats As IConditionalFormats =
worksheet.Range("N6:N35").ConditionalFormats
Dim format As IConditionalFormat = formats.AddCondition()
'Set type as Top for TopBottom rule.
format.FormatType = ExcelCFTType.TopBottom
Dim topBottom As ITopBottom = format.TopBottom
'Set rank value for the TopBottom rule.
topBottom.Type = ExcelCFTopBottomType.Top
'Set rank value for the TopBottom rule.
topBottom.Rank = 10
'Set color for Conditional Formattting.
format.BackColorRGB = System.Drawing.Color.FromArgb(51, 153, 102)
'Saves the Excel
workbook.SaveAs("TopBottom.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile openFile = await openPicker.PickSingleFileAsync();
//Open the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(openFile);
IWorksheet worksheet = workbook.Worksheets[0];
//Applying conditional formatting to "N6:N35".
IConditionalFormats formats = worksheet.Range["N6:N35"].ConditionalFormats;
IConditionalFormat format = formats.AddCondition();
//Applying top or bottom rule in the conditional formatting.
format.FormatType = ExcelCFTType.TopBottom;
ITopBottom topBottom = format.TopBottom;
//Set type as Top for TopBottom rule.
topBottom.Type = ExcelCFTopBottomType.Top;
//Set rank value for the TopBottom rule.
topBottom.Rank = 10;
//Set color for Conditional Formattting.
format.BackColorRGB = System.Drawing.Color.FromArgb(51, 153, 102);
```

```
//Save the workbook
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "TopBottom";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile outputStorageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(outputStorageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("CFTemplate.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "N6:N35".
    IConditionalFormats formats = worksheet.Range["N6:N35"].ConditionalFormats;
    IConditionalFormat format = formats.AddCondition();
    //Applying top or bottom rule in the conditional formatting.
    format.FormatType = ExcelCFTType.TopBottom;
    ITopBottom topBottom = format.TopBottom;
    //Set type as Top for TopBottom rule.
    topBottom.Type = ExcelCFTopBottomType.Top;
    //Set rank value for the TopBottom rule.
    topBottom.Rank = 10;
    //Set color for Conditional Formattting.
    format.BackColorRGB = System.Drawing.Color.FromArgb(51, 153, 102);
    //Saves the excel document to MemoryStream
    FileStream stream = new FileStream("TopBottom.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```






XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.CFT
    emplate.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "N6:N35".
    IConditionalFormats formats = worksheet.Range["N6:N35"].ConditionalFormats;
    IConditionalFormat format = formats.AddCondition();
}
```

```
//Applying top or bottom rule in the conditional formatting.
format.FormatType = ExcelCFType.TopBottom;
ITopBottom topBottom = format.TopBottom;
//Set type as Top for TopBottom rule.
topBottom.Type = ExcelCFTopBottomType.Top;
//Set rank value for the TopBottom rule.
topBottom.Rank = 10;
//Set color for Conditional Formatting.
format.BackColorRGB = System.Drawing.Color.FromArgb(51, 153, 102);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("TopBot
tom.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TopBottom.xlsx",
"application/msexcel", stream);
}
}
```



The following screenshot represents the Excel file generated with TopBottom conditional format with Rank set to 10 in XlsIO.

AutoSave






TopBottom_Bottom_Percent - E...

Sign in




FileHomeInsertDataViewTell me what you want to doShareComments




P38




	A	B	C	D	E	F	G	H	I	J	K	L	M	N		
1	Students Marks Report														Below	Top 10
2	Highlighted by Top 10 / Below Average														Average	Rank
3																
5	Students	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Subject 10	Total	Average	Rank		
6	Student 1	99	97	93	100	91	92	92	95	91	89	939	94	3		
7	Student 2	86	88	90	92	91	87	96	75	80	92	877	88	9		
8	Student 3	91	92	97	94	91	92	95	99	100	94	945	95	1		
9	Student 4	64	71	82	80	63	71	88	78	76	83	756	76	20		
10	Student 5	91	85	79	92	86	81	83	90	82	82	851	85	12		
11	Student 6	90	81	90	83	82	92	95	89	87	88	877	88	9		
12	Student 7	82	89	94	91	86	87	80	83	86	80	858	86	16		
13	Student 8	77	78	60	79	65	77	80	73	70	81	740	74	29		
14	Student 9	71	82	69	75	69	81	70	72	74	84	747	75	27		
15	Student 10	85	81	84	88	83	81	89	88	82	85	846	85	18		
16	Student 11	85	75	79	78	82	86	81	70	79	86	801	80	25		
17	Student 12	91	92	90	100	91	91	92	92	92	91	922	92	4		
18	Student 13	94	81	93	69	82	90	91	92	89	81	862	86	14		
19	Student 14	87	89	87	86	82	80	84	92	90	89	866	87	12		
20	Student 15	78	75	71	75	79	70	72	73	76	77	746	75	28		
21	Student 16	93	91	90	89	82	85	87	88	87	84	876	88	11		
22	Student 17	88	82	80	81	84	81	80	82	91	87	836	84	20		
23	Student 18	93	94	95	92	90	90	98	100	99	94	945	95	1		
24	Student 19	84	90	91	93	90	92	91	81	85	84	881	88	7		
25	Student 20	83	75	74	76	78	89	90	87	82	88	822	82	22		
26	Student 21	89	87	83	80	91	72	92	85	89	94	862	86	14		
27	Student 22	89	87	91	90	90	83	87	91	90	94	892	89	5		
28	Student 23	87	91	90	90	81	79	78	79	80	87	842	84	19		
29	Student 24	79	86	75	75	71	82	85	80	93	94	820	82	23		
30	Student 25	80	79	87	89	91	90	94	93	91	94	888	89	6		
31	Student 26	69	67	71	69	71	69	69	60	71	73	689	69	30		
32	Student 27	73	77	78	82	84	90	91	75	80	85	815	82	24		
33	Student 28	93	87	85	83	71	80	82	85	81	84	831	83	21		
34	Student 29	85	90	93	91	91	92	87	85	85	80	879	88	8		
35	Student 30	89	87	83	86	81	80	92	80	94	92	864	86	13		

Top-Bottom Rules







74%

Note: ITopBottom Rank value should be in a range between 1 and 1000.

Top/Bottom 'n'% rank values

The below code example shows how to format top 50 percentage rank values from the given data range using ITopBottom Type, Rank and Percent properties in XlsIO

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("CFTemplate.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "N6:N35".
    IConditionalFormats formats = worksheet.Range["N6:N35"].ConditionalFormats;
    IConditionalFormat format = formats.AddCondition();
    //Applying top or bottom rule in the conditional formatting.
    format.FormatType = ExcelCFTType.TopBottom;
    ITopBottom topBottom = format.TopBottom;
    //Set type as Bottom for TopBottom rule.
    topBottom.Type = ExcelCFTTopBottomType.Bottom;
    //Set true to Percent property for TopBottom rule.
```

```

topBottom.Percent = true;
//Set rank value for the TopBottom rule.
topBottom.Rank = 50;
//Set color for Conditional Formatting.
format.BackColorRGB = System.Drawing.Color.FromArgb(51, 153, 102);
//Saves the Excel
workbook.SaveAs("TopBottom.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("CFTemplate.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Applying conditional formatting to "N6:N35".
Dim formats As IConditionalFormats =
worksheet.Range("N6:N35").ConditionalFormats
Dim format As IConditionalFormat = formats.AddCondition()
'Set type as Top for TopBottom rule.
format.FormatType = ExcelCFTType.TopBottom
Dim topBottom As ITopBottom = format.TopBottom
'Set type as Bottom for TopBottom rule.
topBottom.Type = ExcelCFTTopBottomType.Bottom
'Set true to Percent property for TopBottom rule.
topBottom.Percent = true
Set rank value for the TopBottom rule.
topBottom.Rank = 50
'Set color for Conditional Formatting.
format.BackColorRGB = System.Drawing.Color.FromArgb(51, 153, 102)
'Saves the Excel
workbook.SaveAs("TopBottom.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile openFile = await openPicker.PickSingleFileAsync();
//Open the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(openFile);
IWorksheet worksheet = workbook.Worksheets[0];
//Applying conditional formatting to "N6:N35".
IConditionalFormats formats = worksheet.Range["N6:N35"].ConditionalFormats;
IConditionalFormat format = formats.AddCondition();
//Applying top or bottom rule in the conditional formatting.
format.FormatType = ExcelCFTType.TopBottom;
ITopBottom topBottom = format.TopBottom;

```

```
//Set type as Bottom for TopBottom rule.
topBottom.Type = ExcelCFTopBottomType.Bottom;
//Set true to Percent property for TopBottom rule.
topBottom.Percent = true;
//Set rank value for the TopBottom rule.
topBottom.Rank = 50;
//Set color for Conditional Formattting.
format.BackColorRGB = System.Drawing.Color.FromArgb(51, 153, 102);
//Save the workbook
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "TopBottom";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile outputStorageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(outputStorageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("CFTemplate.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "N6:N35".
    IConditionalFormats formats = worksheet.Range["N6:N35"].ConditionalFormats;
    IConditionalFormat format = formats.AddCondition();
    //Applying top or bottom rule in the conditional formatting.
    format.FormatType = ExcelCFTType.TopBottom;
    ITopBottom topBottom = format.TopBottom;
    //Set type as Bottom for TopBottom rule.
    topBottom.Type = ExcelCFTopBottomType.Bottom;
    //Set true to Percent property for TopBottom rule.
    topBottom.Percent = true;
    //Set rank value for the TopBottom rule.
    topBottom.Rank = 50;
    //Set color for Conditional Formattting.
    format.BackColorRGB = System.Drawing.Color.FromArgb(51, 153, 102);
    //Saves the excel document to MemoryStream
    FileStream stream = new FileStream("TopBottom.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
```

```

application.DefaultVersion = ExcelVersion.Excel2016;
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.CFT
emplate.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet worksheet = workbook.Worksheets[0];
//Applying conditional formatting to "N6:N35".
IConditionalFormats formats = worksheet.Range["N6:N35"].ConditionalFormats;
IConditionalFormat format = formats.AddCondition();
//Applying top or bottom rule in the conditional formatting.
format.FormatType = ExcelCFTType.TopBottom;
ITopBottom topBottom = format.TopBottom;
//Set type as Bottom for TopBottom rule.
topBottom.Type = ExcelCFTTopBottomType.Bottom;
//Set true to Percent property for TopBottom rule.
topBottom.Percent = true;
//Set rank value for the TopBottom rule.
topBottom.Rank = 50;
//Set color for Conditional Formattting.
format.BackColorRGB = System.Drawing.Color.FromArgb(51, 153, 102);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("TopBot
tom.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TopBottom.xlsx",
"application/msexcel", stream);
}
}

```

The following screenshot represents the Excel file generated with TopBottom conditional format with Percent value set to 50 in XlsIO.

Students Marks Report

Highlighted by Top 10 / Below Average

Students	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Subject 10	Total	Average	Rank
Student 1	99	97	93	100	91	92	92	95	91	89	939	94	3
Student 2	86	88	90	92	91	87	96	75	80	92	877	88	9
Student 3	91	92	97	94	91	92	95	99	100	94	945	95	1
Student 4	64	71	82	80	63	71	88	78	76	83	756	76	20
Student 5	91	85	79	92	86	81	83	90	82	82	851	85	12
Student 6	90	81	90	83	82	92	95	89	87	88	877	88	9
Student 7	82	89	94	91	86	87	80	83	86	80	858	86	10
Student 8	77	78	60	79	65	77	80	73	70	81	740	74	23
Student 9	71	82	69	75	69	81	70	72	74	84	747	75	27
Student 10	85	81	84	88	83	81	89	88	82	85	846	85	18
Student 11	85	75	79	78	82	86	81	70	79	86	801	80	25
Student 12	91	92	90	100	91	91	92	92	92	91	922	92	4
Student 13	94	81	93	69	82	90	91	92	89	81	862	86	14
Student 14	87	89	87	86	82	80	84	92	90	89	866	87	12
Student 15	78	75	71	75	79	70	72	73	76	77	746	75	28
Student 16	93	91	90	89	82	85	87	88	87	84	876	88	11
Student 17	88	82	80	81	84	81	80	82	91	87	836	84	30
Student 18	93	94	95	92	90	90	98	100	99	94	945	95	1
Student 19	84	90	91	93	90	92	91	81	85	84	881	88	7
Student 20	83	75	74	76	78	89	90	87	82	88	822	82	22
Student 21	89	87	83	80	91	72	92	85	89	94	862	86	14
Student 22	89	87	91	90	90	83	87	91	90	94	892	89	5
Student 23	87	91	90	90	81	79	78	79	80	87	842	84	19
Student 24	79	86	75	75	71	82	85	80	93	94	820	82	23
Student 25	80	79	87	89	91	90	94	93	91	94	888	89	6
Student 26	69	67	71	69	71	69	69	60	71	73	689	69	30
Student 27	73	77	78	82	84	90	91	75	80	85	815	82	24
Student 28	93	87	85	83	71	80	82	85	81	84	831	83	21
Student 29	85	90	93	91	91	92	87	85	85	80	879	88	8
Student 30	89	87	83	86	81	80	92	80	94	92	864	86	13

Top-Bottom Rules

Note: ITopBottom Rank value should be in a range between 1 and 100 when set true to Percent property.

Format Above or Below Average Values

Above/Below average rule in conditional formatting is used to highlight the cells which contains above/below the average values in a data range. Top/Bottom conditional formatting rule can be created and customized using the IAboveBelowAverage interface in XlsIO.

The properties of IAboveBelowAverage are:

- **AverageType** - Specifies whether the conditional formatting rule looks for cell values that are above average or below average or standard deviation.
- **StdDevValue** - Specifies standard deviation number for IAboveBelowAverage conditional formatting rule.

The following screenshot represents the input template of conditional formatting.

Students	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Subject 10	Total	Average	Rank
Student 1	99	97	93	100	91	92	92	95	91	89	939	94	3
Student 2	86	88	90	92	91	87	96	75	80	92	877	88	9
Student 3	91	92	97	94	91	92	95	99	100	94	945	95	1
Student 4	64	71	82	80	63	71	88	78	76	83	756	76	26
Student 5	91	85	79	92	86	81	83	90	82	82	851	85	17
Student 6	90	81	90	83	82	92	95	89	87	88	877	88	9
Student 7	82	89	94	91	86	87	80	83	86	80	858	86	16
Student 8	77	78	60	79	65	77	80	73	70	81	740	74	29
Student 9	71	82	69	75	69	81	70	72	74	84	747	75	27
Student 10	85	81	84	88	83	81	89	88	82	85	846	85	18
Student 11	85	75	79	78	82	86	81	70	79	86	801	80	25
Student 12	91	92	90	100	91	91	92	92	92	91	922	92	4
Student 13	94	81	93	69	82	90	91	92	89	81	862	86	14
Student 14	87	89	87	85	82	80	84	92	90	89	866	87	12
Student 15	78	75	71	75	79	70	72	73	76	77	746	75	28
Student 16	93	91	90	89	82	85	87	88	87	84	876	88	11
Student 17	88	82	80	81	84	81	80	82	91	87	836	84	20
Student 18	93	94	95	92	90	90	98	100	99	94	945	95	1
Student 19	84	90	91	93	90	92	91	81	85	84	881	88	7
Student 20	83	75	74	76	78	89	90	87	82	88	822	82	22
Student 21	89	87	83	80	91	72	92	85	89	94	862	86	14
Student 22	89	87	91	90	90	83	87	91	90	94	892	89	5
Student 23	87	91	90	90	81	79	78	79	80	87	842	84	19
Student 24	79	86	75	75	71	82	85	80	93	94	820	82	23
Student 25	80	79	87	89	91	90	94	93	91	94	888	89	6
Student 26	69	67	71	69	71	69	69	60	71	73	689	69	30
Student 27	73	77	78	82	84	90	91	75	80	85	815	82	24
Student 28	93	87	85	83	71	80	82	85	81	84	831	83	21
Student 29	85	90	93	91	91	92	87	85	85	80	879	88	8
Student 30	89	87	83	86	81	80	92	80	94	92	864	86	13

The below code example shows how to format a range with values that are below average using `IAboveBelowAverage` `AverageType` property in XlsIO.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("CFTemplate.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "M6:M35"
    IConditionalFormats formats = worksheet.Range["M6:M35"].ConditionalFormats;
    IConditionalFormat format = formats.AddCondition();
    //Applying above or below average rule in the conditional formatting
    format.FormatType = ExcelCFTType.AboveBelowAverage;
    IAboveBelowAverage aboveBelowAverage = format.AboveBelowAverage;
    //Set AverageType as Below for AboveBelowAverage rule.
    aboveBelowAverage.AverageType = ExcelCFAverageType.Below;
    //Set color for Conditional Formatting.
    format.FontColorRGB = System.Drawing.Color.FromArgb(255, 255, 255);
    format.BackColorRGB = System.Drawing.Color.FromArgb(166, 59, 38);
    //Saves the Excel
    workbook.SaveAs("AboveBelowAverage.xlsx");
}
```



```
}

```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("CFTemplate.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Applying conditional formatting to "M6:M35"
IConditionalFormats formats = worksheet.Range["M6:M35"].ConditionalFormats;
IConditionalFormat format = formats.AddCondition();
'Applying above or below average rule in the conditional formatting
format.FormatType = ExcelCFTType.AboveBelowAverage;
IAboveBelowAverage aboveBelowAverage = format.AboveBelowAverage;
'Set AverageType as Below for AboveBelowAverage rule.
aboveBelowAverage.AverageType = ExcelCFAverageType.Below;
'Set color for Conditional Formattting.
format.FontColorRGB = System.Drawing.Color.FromArgb(255, 255, 255);
format.BackColorRGB = System.Drawing.Color.FromArgb(166, 59, 38);
'Saves the Excel
workbook.SaveAs("AboveBelowAverage.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile openFile = await openPicker.PickSingleFileAsync();
    //Open the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(openFile);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "M6:M35"
    IConditionalFormats formats = worksheet.Range["M6:M35"].ConditionalFormats;
    IConditionalFormat format = formats.AddCondition();
    //Applying above or below average rule in the conditional formatting
    format.FormatType = ExcelCFTType.AboveBelowAverage;
    IAboveBelowAverage aboveBelowAverage = format.AboveBelowAverage;
    //Set AverageType as Below for AboveBelowAverage rule.
    aboveBelowAverage.AverageType = ExcelCFAverageType.Below;
    //Set color for Conditional Formattting.
    format.FontColorRGB = System.Drawing.Color.FromArgb(255, 255, 255);
    format.BackColorRGB = System.Drawing.Color.FromArgb(166, 59, 38);
    //Save the workbook
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "AboveBelowAverage";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
```

```
//Creates a storage file from FileSavePicker
StorageFile outputStorageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(outputStorageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("CFTemplate.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "M6:M35"
    IConditionalFormats formats = worksheet.Range["M6:M35"].ConditionalFormats;
    IConditionalFormat format = formats.AddCondition();
    //Applying above or below average rule in the conditional formatting
    format.FormatType = ExcelCFTType.AboveBelowAverage;
    IAboveBelowAverage aboveBelowAverage = format.AboveBelowAverage;
    //Set AverageType as Below for AboveBelowAverage rule.
    aboveBelowAverage.AverageType = ExcelCFAverageType.Below;
    //Set color for Conditional Formattting.
    format.FontColorRGB = System.Drawing.Color.FromArgb(255, 255, 255);
    format.BackColorRGB = System.Drawing.Color.FromArgb(166, 59, 38);
    //Saves the excel document to MemoryStream
    FileStream stream = new FileStream("AboveBelowAverage.xlsx",
    FileMode.Create, FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.CFT
    emplate.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "M6:M35"
    IConditionalFormats formats = worksheet.Range["M6:M35"].ConditionalFormats;
    IConditionalFormat format = formats.AddCondition();
    //Applying above or below average rule in the conditional formatting
    format.FormatType = ExcelCFTType.AboveBelowAverage;
    IAboveBelowAverage aboveBelowAverage = format.AboveBelowAverage;
    //Set AverageType as Below for AboveBelowAverage rule.
    aboveBelowAverage.AverageType = ExcelCFAverageType.Below;
    //Set color for Conditional Formattting.
    format.FontColorRGB = System.Drawing.Color.FromArgb(255, 255, 255);
}
```



```
format.BackColorRGB = System.Drawing.Color.FromArgb(166, 59, 38);  
//Saving the workbook as stream  
MemoryStream stream = new MemoryStream();  
workbook.SaveAs(stream);  
stream.Position = 0;  
//Save the document as file and view the saved document  
//The operation in SaveAndView under Xamarin varies between Windows Phone,  
//Android and iOS platforms. Please refer xlsio/xamarin section for respective  
//code samples.  
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==  
TargetPlatform.Windows)  
{  
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("AboveB  
elowAverage.xlsx", "application/msexcel", stream);  
}  
else  
{  
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("AboveBelowAverage.  
xlsx", "application/msexcel", stream);  
}  
}
```

The following screenshot represents the Excel file generated with AboveBelowAverage conditional format with AverageType set as Below in XlsIO.

The screenshot shows an Excel spreadsheet titled "Students Marks Report". The spreadsheet has columns for Student names, 10 subjects, Total, Average, and Rank. The 'Average' column is highlighted with conditional formatting, showing values above and below the average. The 'Rank' column is also highlighted with conditional formatting, showing values above and below the average.

Students	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Subject 10	Total	Average	Rank
Student 1	99	97	93	100	91	92	92	95	91	89	939	94	3
Student 2	86	88	90	92	91	87	96	75	80	92	877	88	9
Student 3	91	92	97	94	91	92	95	99	100	94	945	95	1
Student 4	64	71	82	80	63	71	88	78	76	83	756	76	26
Student 5	91	85	79	92	86	81	83	90	82	82	851	85	17
Student 6	90	81	90	83	82	92	95	89	87	88	877	88	9
Student 7	82	89	94	91	86	87	80	83	86	80	858	86	16
Student 8	77	78	60	79	65	77	80	73	70	81	740	74	29
Student 9	71	82	69	75	69	81	70	72	74	84	747	75	27
Student 10	85	81	84	88	83	81	89	88	82	85	846	85	18
Student 11	85	75	79	78	82	86	81	70	79	86	801	80	25
Student 12	91	92	90	100	91	91	92	92	92	91	922	92	4
Student 13	94	81	93	69	82	90	91	92	89	81	862	86	14
Student 14	87	89	87	86	82	80	84	92	90	89	866	87	12
Student 15	78	75	71	75	79	70	72	73	76	77	746	75	28
Student 16	93	91	90	89	82	85	87	88	87	84	876	88	11
Student 17	88	82	80	81	84	81	80	82	91	87	836	84	20
Student 18	93	94	95	92	90	90	98	100	99	94	945	95	1
Student 19	84	90	91	93	90	92	91	81	85	84	881	88	7
Student 20	83	75	74	76	78	89	90	87	82	88	822	82	22
Student 21	89	87	83	80	91	72	92	85	89	94	862	86	14
Student 22	89	87	91	90	90	83	87	91	90	94	892	89	5
Student 23	87	91	90	90	81	79	78	79	80	87	842	84	19
Student 24	79	86	75	75	71	82	85	80	93	94	820	82	23
Student 25	80	79	87	89	91	90	94	93	91	94	888	89	6
Student 26	69	67	71	69	71	69	69	60	71	73	689	69	30
Student 27	73	77	78	82	84	90	91	75	80	85	815	82	24
Student 28	93	87	85	83	71	80	82	85	81	84	831	83	21
Student 29	85	90	93	91	91	92	87	85	85	80	879	88	8
Student 30	89	87	83	86	81	80	92	80	94	92	864	86	13

Above or Below Standard Deviation values

The below code example shows how to format a range with values above standard deviation, using `IAboveBelowAverage` `AverageType` and `StdDevValue` properties in XlsIO.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("CFTemplate.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "M6:M35"
    IConditionalFormats formats = worksheet.Range["M6:M35"].ConditionalFormats;
    IConditionalFormat format = formats.AddCondition();
    //Applying above or below average rule in the conditional formatting
    format.FormatType = ExcelCFTType.AboveBelowAverage;
    IAboveBelowAverage aboveBelowAverage = format.AboveBelowAverage;
    //Set AverageType as AboveStdDev for AboveBelowAverage rule.
    aboveBelowAverage.AverageType = ExcelCFAverageType.AboveStdDev;
    //Set value to StdDevValue property for AboveBelowAverage rule.
    aboveBelowAverage.StdDevValue = 1;
    //Set color for Conditional Formatting.
}
```

```
format.FontColorRGB = System.Drawing.Color.FromArgb(255, 255, 255);
format.BackColorRGB = System.Drawing.Color.FromArgb(166, 59, 38);
//Saves the Excel
workbook.SaveAs ("AboveBelowAverage.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("CFTemplate.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Applying conditional formatting to "M6:M35"
IConditionalFormats formats = worksheet.Range["M6:M35"].ConditionalFormats;
IConditionalFormat format = formats.AddCondition();
'Applying above or below average rule in the conditional formatting
format.FormatType = ExcelCFTType.AboveBelowAverage;
IAboveBelowAverage aboveBelowAverage = format.AboveBelowAverage;
'Set AverageType as AboveStdDev for AboveBelowAverage rule.
aboveBelowAverage.AverageType = ExcelCFAverageType.AboveStdDev
'Set value to StdDevValue property for AboveBelowAverage rule.
aboveBelowAverage.StdDevValue = 1
'Set color for Conditional Formattting.
format.FontColorRGB = System.Drawing.Color.FromArgb(255, 255, 255);
format.BackColorRGB = System.Drawing.Color.FromArgb(166, 59, 38);
'Saves the Excel
workbook.SaveAs ("AboveBelowAverage.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile openFile = await openPicker.PickSingleFileAsync();
//Open the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(openFile);
IWorksheet worksheet = workbook.Worksheets[0];
//Applying conditional formatting to "M6:M35"
IConditionalFormats formats = worksheet.Range["M6:M35"].ConditionalFormats;
IConditionalFormat format = formats.AddCondition();
//Applying above or below average rule in the conditional formatting
format.FormatType = ExcelCFTType.AboveBelowAverage;
IAboveBelowAverage aboveBelowAverage = format.AboveBelowAverage;
//Set AverageType as AboveStdDev for AboveBelowAverage rule.
aboveBelowAverage.AverageType = ExcelCFAverageType.AboveStdDev;
//Set value to StdDevValue property for AboveBelowAverage rule.
aboveBelowAverage.StdDevValue = 1;
//Set color for Conditional Formattting.
```

```

format.FontColorRGB = System.Drawing.Color.FromArgb(255, 255, 255);
format.BackColorRGB = System.Drawing.Color.FromArgb(166, 59, 38);
//Save the workbook
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "AboveBelowAverage";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile outputStorageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(outputStorageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("CFTemplate.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Applying conditional formatting to "M6:M35"
    IConditionalFormats formats = worksheet.Range["M6:M35"].ConditionalFormats;
    IConditionalFormat format = formats.AddCondition();
    //Applying above or below average rule in the conditional formatting
    format.FormatType = ExcelCFTType.AboveBelowAverage;
    IAboveBelowAverage aboveBelowAverage = format.AboveBelowAverage;
    //Set AverageType as AboveStdDev for AboveBelowAverage rule.
    aboveBelowAverage.AverageType = ExcelCFAverageType.AboveStdDev;
    //Set value to StdDevValue property for AboveBelowAverage rule.
    aboveBelowAverage.StdDevValue = 1;
    //Set color for Conditional Formatting.
    format.FontColorRGB = System.Drawing.Color.FromArgb(255, 255, 255);
    format.BackColorRGB = System.Drawing.Color.FromArgb(166, 59, 38);
    //Saves the excel document to MemoryStream
    FileStream stream = new FileStream("AboveBelowAverage.xlsx",
    FileMode.Create, FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.CFT
    emplate.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
}

```

```

//Applying conditional formatting to "M6:M35"
IConditionalFormats formats = worksheet.Range["M6:M35"].ConditionalFormats;
IConditionalFormat format = formats.AddCondition();
//Applying above or below average rule in the conditional formatting
format.FormatType = ExcelCFType.AboveBelowAverage;
IAboveBelowAverage aboveBelowAverage = format.AboveBelowAverage;
//Set AverageType as AboveStdDev for AboveBelowAverage rule.
aboveBelowAverage.AverageType = ExcelCFAverageType.AboveStdDev;
//Set value to StdDevValue property for AboveBelowAverage rule.
aboveBelowAverage.StdDevValue = 1;
//Set color for Conditional Formattting.
format.FontColorRGB = System.Drawing.Color.FromArgb(255, 255, 255);
format.BackColorRGB = System.Drawing.Color.FromArgb(166, 59, 38);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("AboveB
elowAverage.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("AboveBelowAverage.
xlsx", "application/msexcel", stream);
}
}

```

The following screenshot represents the Excel file generated with **AboveBelowAverage** conditional format when **AverageType** is set as **AboveStdDev** in XlsIO.

Students	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Subject 10	Total	Average	Rank
Student 1	99	97	93	100	91	92	92	95	91	89	939	94	3
Student 2	86	88	90	92	91	87	96	75	80	92	877	88	9
Student 3	91	92	97	94	91	92	95	99	100	94	945	95	1
Student 4	64	71	82	80	63	71	88	78	76	83	756	76	26
Student 5	91	85	79	92	86	81	83	90	82	82	851	85	17
Student 6	90	81	90	83	82	92	95	89	87	88	877	88	9
Student 7	82	89	94	91	86	87	80	83	86	80	858	86	16
Student 8	77	78	60	79	65	77	80	73	70	81	740	74	29
Student 9	71	82	69	75	69	81	70	72	74	84	747	75	27
Student 10	85	81	84	88	83	81	89	88	82	85	846	85	18
Student 11	85	75	79	78	82	86	81	70	79	86	801	80	25
Student 12	91	92	90	100	91	91	92	92	92	91	922	92	4
Student 13	94	81	93	69	82	90	91	92	89	81	862	86	14
Student 14	87	89	87	86	82	80	84	92	90	89	866	87	12
Student 15	78	75	71	75	79	70	72	73	76	77	746	75	28
Student 16	93	91	90	89	82	85	87	88	87	84	876	88	11
Student 17	88	82	80	81	84	81	80	82	91	87	836	84	20
Student 18	93	94	95	92	90	90	98	100	99	94	945	95	1
Student 19	84	90	91	93	90	92	91	81	85	84	881	88	7
Student 20	83	75	74	76	78	89	90	87	82	88	822	82	22
Student 21	89	87	83	80	91	72	92	85	89	94	862	86	14
Student 22	89	87	91	90	90	83	87	91	90	94	892	89	5
Student 23	87	91	90	90	81	79	78	79	80	87	842	84	19
Student 24	79	86	75	75	71	82	85	80	93	94	820	82	23
Student 25	80	79	87	89	91	90	94	93	91	94	888	89	6
Student 26	69	67	71	69	71	69	69	60	71	73	689	69	30
Student 27	73	77	78	82	84	90	91	75	80	85	815	82	24
Student 28	93	87	85	83	71	80	82	85	81	84	831	83	21
Student 29	85	90	93	91	91	92	87	85	85	80	879	88	8
Student 30	89	87	83	86	81	80	92	80	94	92	864	86	13

Note: `IAboveBelowAverage StdDevValue` can be applied only if the `AverageType` is `AboveStdDev` or `BelowStdDev`. The `StdDevValue` value should be in a range between 1 and 3.

Advanced Conditional Format Types

In conjunction with basic conditional formatting, the new formatting visualizations such as **Data Bars**, **Color Scales**, and **Icon Sets** are supported in XlsIO.

Data Bars

Here, the values in each of the selected cells are compared, and a data bar is drawn in each cell representing the value of that cell relative to the other cells in the selected range. This bar provides a clear visual cue for users, making it easier to pick out larger and smaller values in a range.

This can be set and manipulated using the `IDataBar` interface as follows.

C#

```
//Create data bars for the data in specified range
IConditionalFormats conditionalFormats =
worksheet.Range["C7:C46"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCFTType.DataBar;
IDataBar dataBar = conditionalFormat.DataBar;
//Set the constraints
dataBar.MinPoint.Type = ConditionValueType.LowestValue;
```

```
dataBar.MaxPoint.Type = ConditionValueType.HighestValue;
//Set color for DataBar
dataBar.BarColor = Color.FromArgb(156, 208, 243);
//Hide the data bar values
dataBar.ShowValue = false;
dataBar.BarColor = Color.Aqua;
```

VB.NET

```
'Create data bars for the data in specified range
Dim formats As IConditionalFormats =
worksheet.Range("C7:C46").ConditionalFormats
Dim format As IConditionalFormat = formats.AddCondition()
format.FormatType = ExcelCfType.DataBar
Dim dataBar As IDataBar = format.DataBar
'Set the constraints
dataBar.MinPoint.Type = ConditionValueType.LowestValue
dataBar.MaxPoint.Type = ConditionValueType.HighestValue
'Set color for DataBar
dataBar.BarColor = System.Drawing.Color.FromArgb(156, 208, 243)
'Hide the data bar values
dataBar.ShowValue = False
dataBar.BarColor = Color.Aqua
```

UWP

```
//Create data bars for the data in specified range
IConditionalFormats conditionalFormats =
worksheet.Range["C7:C46"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCfType.DataBar;
IDataBar dataBar = conditionalFormat.DataBar;
//Set the constraints
dataBar.MinPoint.Type = ConditionValueType.LowestValue;
dataBar.MaxPoint.Type = ConditionValueType.HighestValue;
//Set color for DataBar
dataBar.BarColor = Color.FromArgb(255, 156, 208, 243);
//Hide the data bar values
dataBar.ShowValue = false;
dataBar.BarColor = Color.FromArgb(255, 0, 255, 255);
```

ASP.NET CORE

```
//Create data bars for the data in specified range
IConditionalFormats conditionalFormats =
worksheet.Range["C7:C46"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCfType.DataBar;
IDataBar dataBar = conditionalFormat.DataBar;
//Set the constraints
dataBar.MinPoint.Type = ConditionValueType.LowestValue;
dataBar.MaxPoint.Type = ConditionValueType.HighestValue;
//Set color for DataBar
dataBar.BarColor = Color.FromArgb(156, 208, 243);
//Hide the data bar values
```

```
dataBar.ShowValue = false;
dataBar.BarColor = Color.Aqua;
```

XAMARIN

```
//Create data bars for the data in specified range
IConditionalFormats conditionalFormats =
worksheet.Range["C7:C46"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCFTType.DataBar;
IDataBar dataBar = conditionalFormat.DataBar;
//Set the constraints
dataBar.MinPoint.Type = ConditionValueType.LowestValue;
dataBar.MaxPoint.Type = ConditionValueType.HighestValue;
//Set color for DataBar
dataBar.BarColor = Syncfusion.Drawing.Color.FromArgb(255, 156, 208, 243);
//Hide the data bar values
dataBar.ShowValue = false;
dataBar.BarColor = Color.Aqua;
```

Color Scales

Color Scales let you create visual effects in your data to see how the value of a cell is compared with the values in a range of cells. A color scale uses cell shading, as opposed to bars, to communicate relative values, beyond the relative size of the value of a cell.

Creation of color scales and its formatting rules using the **IColorScale** interface in XlsIO is illustrated as follows.

C#

```
//Create color scale for the data in specified range
IConditionalFormats conditionalFormats =
worksheet.Range["D7:D46"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCFTType.ColorScale;
IColorScale colorScale = conditionalFormat.ColorScale;
//Sets 3 - color scale and its constraints
colorScale.SetConditionCount(3);
colorScale.Criteria[0].FormatColorRGB = Color.FromArgb(230, 197, 218);
colorScale.Criteria[0].Type = ConditionValueType.LowestValue;
colorScale.Criteria[0].Value = "0";
colorScale.Criteria[1].FormatColorRGB = Color.FromArgb(244, 210, 178);
colorScale.Criteria[1].Type = ConditionValueType.Percentile;
colorScale.Criteria[1].Value = "50";
colorScale.Criteria[2].FormatColorRGB = Color.FromArgb(245, 247, 171);
colorScale.Criteria[2].Type = ConditionValueType.HighestValue;
colorScale.Criteria[2].Value = "0";
conditionalFormat.FirstFormulaR1C1 = "=R[1]C[0]";
conditionalFormat.SecondFormulaR1C1 = "=R[1]C[1]";
```

VB.NET

```
'Create color scale for the data in specified range
Dim conditionalFormats As IConditionalFormats =
worksheet.Range("D7:D46").ConditionalFormats
```



```

Dim conditionalFormat As IConditionalFormat =
conditionalFormats.AddCondition()
conditionalFormat.FormatType = ExcelCfType.ColorScale
Dim colorScale As IColorScale = conditionalFormat.ColorScale
'Sets 3 - color scale and its constraints
colorScale.SetConditionCount(3)
colorScale.Criteria(0).FormatColorRGB = System.Drawing.Color.FromArgb(230,
197, 218)
colorScale.Criteria(0).Type = ConditionValueType.LowestValue
colorScale.Criteria(0).Value = "0"
colorScale.Criteria(1).FormatColorRGB = System.Drawing.Color.FromArgb(244,
210, 178)
colorScale.Criteria(1).Type = ConditionValueType.Percentile
colorScale.Criteria(1).Value = "50"
colorScale.Criteria(2).FormatColorRGB = System.Drawing.Color.FromArgb(245,
247, 171)
colorScale.Criteria(2).Type = ConditionValueType.HighestValue
colorScale.Criteria(2).Value = "0"

```

UWP

```

//Create color scale for the data in specified range
IConditionalFormats conditionalFormats =
worksheet.Range["D7:D46"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCfType.ColorScale;
IColorScale colorScale = conditionalFormat.ColorScale;
//Sets 3 - color scale and its constraints
colorScale.SetConditionCount(3);
colorScale.Criteria[0].FormatColorRGB = Color.FromArgb(255,230, 197, 218);
colorScale.Criteria[0].Type = ConditionValueType.LowestValue;
colorScale.Criteria[0].Value = "0";
colorScale.Criteria[1].FormatColorRGB = Color.FromArgb(255,244, 210, 178);
colorScale.Criteria[1].Type = ConditionValueType.Percentile;
colorScale.Criteria[1].Value = "50";
colorScale.Criteria[2].FormatColorRGB = Color.FromArgb(255,245, 247, 171);
colorScale.Criteria[2].Type = ConditionValueType.HighestValue;
colorScale.Criteria[2].Value = "0";
conditionalFormat.FirstFormulaR1C1 = "=R[1]C[0]";
conditionalFormat.SecondFormulaR1C1 = "=R[1]C[1]";

```

ASP.NET CORE

```

//Create color scale for the data in specified range
IConditionalFormats conditionalFormats =
worksheet.Range["D7:D46"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCfType.ColorScale;
IColorScale colorScale = conditionalFormat.ColorScale;
//Sets 3 - color scale and its constraints
colorScale.SetConditionCount(3);
colorScale.Criteria[0].FormatColorRGB = Color.FromArgb(230, 197, 218);
colorScale.Criteria[0].Type = ConditionValueType.LowestValue;
colorScale.Criteria[0].Value = "0";
colorScale.Criteria[1].FormatColorRGB = Color.FromArgb(244, 210, 178);
colorScale.Criteria[1].Type = ConditionValueType.Percentile;

```

```
colorScale.Criteria[1].Value = "50";
colorScale.Criteria[2].FormatColorRGB = Color.FromArgb(245, 247, 171);
colorScale.Criteria[2].Type = ConditionValueType.HighestValue;
colorScale.Criteria[2].Value = "0";
conditionalFormat.FirstFormulaR1C1 = "=R[1]C[0]";
conditionalFormat.SecondFormulaR1C1 = "=R[1]C[1]";
```

XAMARIN

```
//Create color scale for the data in specified range
IConditionalFormats conditionalFormats =
worksheet.Range["D7:D46"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCfType.ColorScale;
IColorScale colorScale = conditionalFormat.ColorScale;
//Sets 3 - color scale and its constraints
colorScale.SetConditionCount(3);
colorScale.Criteria[0].FormatColorRGB =
Syncfusion.Drawing.Color.FromArgb(230, 197, 218);
colorScale.Criteria[0].Type = ConditionValueType.LowestValue;
colorScale.Criteria[0].Value = "0";
colorScale.Criteria[1].FormatColorRGB =
Syncfusion.Drawing.Color.FromArgb(244, 210, 178);
colorScale.Criteria[1].Type = ConditionValueType.Percentile;
colorScale.Criteria[1].Value = "50";
colorScale.Criteria[2].FormatColorRGB =
Syncfusion.Drawing.Color.FromArgb(245, 247, 171);
colorScale.Criteria[2].Type = ConditionValueType.HighestValue;
colorScale.Criteria[2].Value = "0";
conditionalFormat.FirstFormulaR1C1 = "=R[1]C[0]";
conditionalFormat.SecondFormulaR1C1 = "=R[1]C[1]";
```

Icon Sets

Icon sets present data in three to five categories that are distinguished by a threshold value. Each icon represents a range of values and each cell is annotated with the icon that represents that range.

Icon sets can be created and customized in XlsIO as follows.

C#

```
//Create icon sets for the data in specified range
IConditionalFormats conditionalFormats =
worksheet.Range["E7:E46"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCfType.IconSet;
IIconSet iconSet = conditionalFormat.IconSet;
//Apply three symbols icon and hide the data in the specified range
iconSet.IconSet = ExcelIconSetType.ThreeSymbols;
iconSet.IconCriteria[1].Type = ConditionValueType.Percent;
iconSet.IconCriteria[1].Value = "50";
iconSet.IconCriteria[2].Type = ConditionValueType.Percent;
iconSet.IconCriteria[2].Value = "50";
iconSet.ShowIconOnly = true;
```

VB.NET

```

'Create icon sets for the data in specified range
Dim conditionalFormats As IConditionalFormats =
worksheet.Range("E7:E46").ConditionalFormats
Dim conditionalFormat As IConditionalFormat = formats.AddCondition()
format.FormatType = ExcelCfType.IconSet
Dim iconSet As IIconSet = format.IconSet
'Apply three symbols icon and hide the data in the specified range
iconSet.IconSet = ExcelIconSetType.ThreeSymbols
iconSet.IconCriteria(1).Type = ConditionValueType.Percent
iconSet.IconCriteria(1).Value = "50"
iconSet.IconCriteria(2).Type = ConditionValueType.Percent
iconSet.IconCriteria(2).Value = "50"
iconSet.ShowIconOnly = True

```

UWP

```

//Create icon sets for the data in specified range
IConditionalFormats conditionalFormats =
worksheet.Range["E7:E46"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCfType.IconSet;
IIconSet iconSet = conditionalFormat.IconSet;
//Apply three symbols icon and hide the data in the specified range
iconSet.IconSet = ExcelIconSetType.ThreeSymbols;
iconSet.IconCriteria[1].Type = ConditionValueType.Percent;
iconSet.IconCriteria[1].Value = "50";
iconSet.IconCriteria[2].Type = ConditionValueType.Percent;
iconSet.IconCriteria[2].Value = "50";
iconSet.ShowIconOnly = true;

```

ASP.NET CORE

```

//Create icon sets for the data in specified range
IConditionalFormats conditionalFormats =
worksheet.Range["E7:E46"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCfType.IconSet;
IIconSet iconSet = conditionalFormat.IconSet;
//Apply three symbols icon and hide the data in the specified range
iconSet.IconSet = ExcelIconSetType.ThreeSymbols;
iconSet.IconCriteria[1].Type = ConditionValueType.Percent;
iconSet.IconCriteria[1].Value = "50";
iconSet.IconCriteria[2].Type = ConditionValueType.Percent;
iconSet.IconCriteria[2].Value = "50";
iconSet.ShowIconOnly = true;

```

XAMARIN

```

//Create icon sets for the data in specified range
IConditionalFormats conditionalFormats =
worksheet.Range["E7:E46"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCfType.IconSet;
IIconSet iconSet = conditionalFormat.IconSet;
//Apply three symbols icon and hide the data in the specified range

```

```

iconSet.IconSet = ExcelIconSetType.ThreeSymbols;
iconSet.IconCriteria[1].Type = ConditionValueType.Percent;
iconSet.IconCriteria[1].Value = "50";
iconSet.IconCriteria[2].Type = ConditionValueType.Percent;
iconSet.IconCriteria[2].Value = "50";
iconSet.ShowIconOnly = true;

```

Custom Icon Sets

You can customize the icon set by changing the IconSet and Index properties for each icon criteria.

Custom Icon sets can be created and customized in XlsIO as follows.

C#

```

// Create icon sets for the data in specified range
IConditionalFormats conditionalFormats =
sheet.Range["H1:K6"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCFTType.IconSet;
IIIconSet iconSet = conditionalFormat.IconSet;
iconSet.IconSet = ExcelIconSetType.ThreeFlags;
IIIconConditionValue iconValue1 = iconSet.IconCriteria[0] as
IIIconConditionValue;
iconValue1.IconSet = ExcelIconSetType.FiveBoxes;
iconValue1.Index = 3;
iconValue1.Type = ConditionValueType.Percent;
iconValue1.Value = "25";
iconValue1.Operator = ConditionalFormatOperator.GreaterThan;
IIIconConditionValue iconValue2 = iconSet.IconCriteria[1] as
IIIconConditionValue;
iconValue2.IconSet = ExcelIconSetType.ThreeSigns;
iconValue2.Index = 2;
iconValue2.Type = ConditionValueType.Percent;
iconValue2.Value = "50";
iconValue2.Operator = ConditionalFormatOperator.GreaterThan;
IIIconConditionValue iconValue3 = iconSet.IconCriteria[2] as
IIIconConditionValue;
iconValue3.IconSet = ExcelIconSetType.FourRating;
iconValue3.Index = 0;
iconValue3.Type = ConditionValueType.Percent;
iconValue3.Value = "75";
iconValue3.Operator = ConditionalFormatOperator.GreaterThan;

```

VB.NET

```

Dim conditionalFormats As IConditionalFormats =
sheet.Range("H1:K6").ConditionalFormats
Dim conditionalFormat As IConditionalFormat =
conditionalFormats.AddCondition
conditionalFormat.FormatType = ExcelCFTType.IconSet
Dim iconSet As IIIconSet = conditionalFormat.IconSet
iconSet.IconSet = ExcelIconSetType.ThreeFlags
Dim iconValue1 As IIIconConditionValue =
CType(iconSet.IconCriteria(0), IIIconConditionValue)
iconValue1.IconSet = ExcelIconSetType.FiveBoxes
iconValue1.Index = 3

```

```

iconValue1.Type = ConditionValueType.Percent
iconValue1.Value = "25"
iconValue1.Operator = ConditionalFormatOperator.GreaterThan
Dim iconValue2 As IIconConditionValue =
CType(iconSet.IconCriteria(1), IIconConditionValue)
iconValue2.IconSet = ExcelIconSetType.ThreeSigns
iconValue2.Index = 2
iconValue2.Type = ConditionValueType.Percent
iconValue2.Value = "50"
iconValue2.Operator = ConditionalFormatOperator.GreaterThan
Dim iconValue3 As IIconConditionValue =
CType(iconSet.IconCriteria(2), IIconConditionValue)
iconValue3.IconSet = ExcelIconSetType.FourRating
iconValue3.Index = 0
iconValue3.Type = ConditionValueType.Percent
iconValue3.Value = "75"
iconValue3.Operator = ConditionalFormatOperator.GreaterThan

```

UWP

```

// Create icon sets for the data in specified range
IConditionalFormats conditionalFormats =
sheet.Range["H1:K6"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCFTType.IconSet;
IIconSet iconSet = conditionalFormat.IconSet;
iconSet.IconSet = ExcelIconSetType.ThreeFlags;
IIconConditionValue iconValue1 = iconSet.IconCriteria[0] as
IIconConditionValue;
iconValue1.IconSet = ExcelIconSetType.FiveBoxes;
iconValue1.Index = 3;
iconValue1.Type = ConditionValueType.Percent;
iconValue1.Value = "25";
iconValue1.Operator = ConditionalFormatOperator.GreaterThan;
IIconConditionValue iconValue2 = iconSet.IconCriteria[1] as
IIconConditionValue;
iconValue2.IconSet = ExcelIconSetType.ThreeSigns;
iconValue2.Index = 2;
iconValue2.Type = ConditionValueType.Percent;
iconValue2.Value = "50";
iconValue2.Operator = ConditionalFormatOperator.GreaterThan;
IIconConditionValue iconValue3 = iconSet.IconCriteria[2] as
IIconConditionValue;
iconValue3.IconSet = ExcelIconSetType.FourRating;
iconValue3.Index = 0;
iconValue3.Type = ConditionValueType.Percent;
iconValue3.Value = "75";
iconValue3.Operator = ConditionalFormatOperator.GreaterThan;

```

ASP.NET CORE

```

// Create icon sets for the data in specified range
IConditionalFormats conditionalFormats =
sheet.Range["H1:K6"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCFTType.IconSet;

```

```

IIconSet iconSet = conditionalFormat.IconSet;
iconSet.IconSet = ExcelIconSetType.ThreeFlags;
IIconConditionValue iconValue1 = iconSet.IconCriteria[0] as
IIconConditionValue;
iconValue1.IconSet = ExcelIconSetType.FiveBoxes;
iconValue1.Index = 3;
iconValue1.Type = ConditionValueType.Percent;
iconValue1.Value = "25";
iconValue1.Operator = ConditionalFormatOperator.GreaterThan;
IIconConditionValue iconValue2 = iconSet.IconCriteria[1] as
IIconConditionValue;
iconValue2.IconSet = ExcelIconSetType.ThreeSigns;
iconValue2.Index = 2;
iconValue2.Type = ConditionValueType.Percent;
iconValue2.Value = "50";
iconValue2.Operator = ConditionalFormatOperator.GreaterThan;
IIconConditionValue iconValue3 = iconSet.IconCriteria[2] as
IIconConditionValue;
iconValue3.IconSet = ExcelIconSetType.FourRating;
iconValue3.Index = 0;
iconValue3.Type = ConditionValueType.Percent;
iconValue3.Value = "75";
iconValue3.Operator = ConditionalFormatOperator.GreaterThan;

```

XAMARIN

```

// Create icon sets for the data in specified range
IConditionalFormats conditionalFormats =
sheet.Range["H1:K6"].ConditionalFormats;
IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCFTType.IconSet;
IIconSet iconSet = conditionalFormat.IconSet;
iconSet.IconSet = ExcelIconSetType.ThreeFlags;
IIconConditionValue iconValue1 = iconSet.IconCriteria[0] as
IIconConditionValue;
iconValue1.IconSet = ExcelIconSetType.FiveBoxes;
iconValue1.Index = 3;
iconValue1.Type = ConditionValueType.Percent;
iconValue1.Value = "25";
iconValue1.Operator = ConditionalFormatOperator.GreaterThan;
IIconConditionValue iconValue2 = iconSet.IconCriteria[1] as
IIconConditionValue;
iconValue2.IconSet = ExcelIconSetType.ThreeSigns;
iconValue2.Index = 2;
iconValue2.Type = ConditionValueType.Percent;
iconValue2.Value = "50";
iconValue2.Operator = ConditionalFormatOperator.GreaterThan;
IIconConditionValue iconValue3 = iconSet.IconCriteria[2] as
IIconConditionValue;
iconValue3.IconSet = ExcelIconSetType.FourRating;
iconValue3.Index = 0;
iconValue3.Type = ConditionValueType.Percent;
iconValue3.Value = "75";
iconValue3.Operator = ConditionalFormatOperator.GreaterThan;

```

The application of these visualizations to a sample data and its output file is represented in the following code example.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("SampleTemplate.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create data bars for the data in specified range
    IConditionalFormats conditionalFormats =
        worksheet.Range["C7:C46"].ConditionalFormats;
    IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
    conditionalFormat.FormatType = ExcelCFType.DataBar;
    IDataBar dataBar = conditionalFormat.DataBar;
    //Set the constraints
    dataBar.MinPoint.Type = ConditionValueType.LowestValue;
    dataBar.MaxPoint.Type = ConditionValueType.HighestValue;
    //Set color for Bar
    dataBar.BarColor = Color.FromArgb(156, 208, 243);
    //Hide the values in data bar
    dataBar.ShowValue = false;
    dataBar.BarColor = Color.Aqua;
    //Create color scales for the data in specified range
    conditionalFormats = worksheet.Range["D7:D46"].ConditionalFormats;
    conditionalFormat = conditionalFormats.AddCondition();
    conditionalFormat.FormatType = ExcelCFType.ColorScale;
    IColorScale colorScale = conditionalFormat.ColorScale;
    //Sets 3 - color scale
    colorScale.SetConditionCount(3);
    colorScale.Criteria[0].FormatColorRGB = Color.FromArgb(230, 197, 218);
    colorScale.Criteria[0].Type = ConditionValueType.LowestValue;
    colorScale.Criteria[0].Value = "0";
    colorScale.Criteria[1].FormatColorRGB = Color.FromArgb(244, 210, 178);
    colorScale.Criteria[1].Type = ConditionValueType.Percentile;
    colorScale.Criteria[1].Value = "50";
    colorScale.Criteria[2].FormatColorRGB = Color.FromArgb(245, 247, 171);
    colorScale.Criteria[2].Type = ConditionValueType.HighestValue;
    colorScale.Criteria[2].Value = "0";
    conditionalFormat.FirstFormulaR1C1 = "=R[1]C[0]";
    conditionalFormat.SecondFormulaR1C1 = "=R[1]C[1]";
    //Create icon sets for the data in specified range
    conditionalFormats = worksheet.Range["E7:E46"].ConditionalFormats;
    conditionalFormat = conditionalFormats.AddCondition();
    conditionalFormat.FormatType = ExcelCFType.IconSet;
    IIconSet iconSet = conditionalFormat.IconSet;
    //Apply three symbols icon and hide the data in the specified range
    iconSet.IconSet = ExcelIconSetType.ThreeSymbols;
    iconSet.IconCriteria[1].Type = ConditionValueType.Percent;
    iconSet.IconCriteria[1].Value = "50";
    iconSet.IconCriteria[2].Type = ConditionValueType.Percent;
    iconSet.IconCriteria[2].Value = "50";
    iconSet.ShowIconOnly = true;
    string fileName = "ConditionalFormatting.xlsx";
    workbook.SaveAs(fileName);
}
```

```
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook =
application.Workbooks.Open("SampleTemplate.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Create data bars for the data in specified range
Dim formats As IConditionalFormats =
worksheet.Range("C7:C46").ConditionalFormats
Dim format As IConditionalFormat = formats.AddCondition()
format.FormatType = ExcelCfType.DataBar
Dim dataBar As IDataBar = format.DataBar
'Set the constraints
dataBar.MinPoint.Type = ConditionValueType.LowestValue
dataBar.MaxPoint.Type = ConditionValueType.HighestValue
'Set color for Bar
dataBar.BarColor = System.Drawing.Color.FromArgb(156, 208, 243)
'Hide the value in data bar
dataBar.ShowValue = False
dataBar.BarColor = Color.Aqua
'Create color scales for the data in specified range
formats = worksheet.Range("D7:D46").ConditionalFormats
format = formats.AddCondition()
format.FormatType = ExcelCfType.ColorScale
Dim colorScale As IColorScale = format.ColorScale
'Sets 3 - color scale
colorScale.SetConditionCount(3)
colorScale.Criteria(0).FormatColorRGB = Color.FromArgb(230, 197, 218)
colorScale.Criteria(0).Type = ConditionValueType.LowestValue
colorScale.Criteria(0).Value = "0"
colorScale.Criteria(1).FormatColorRGB = Color.FromArgb(244, 210, 178)
colorScale.Criteria(1).Type = ConditionValueType.Percentile
colorScale.Criteria(1).Value = "50"
colorScale.Criteria(2).FormatColorRGB = Color.FromArgb(245, 247, 171)
colorScale.Criteria(2).Type = ConditionValueType.HighestValue
colorScale.Criteria(2).Value = "0"
'Create icon sets for the data in specified range
formats = worksheet.Range("E7:E46").ConditionalFormats
format = formats.AddCondition()
format.FormatType = ExcelCfType.IconSet
Dim iconSet As IIconSet = format.IconSet
'Apply three symbols icon and hide the data in the specified range
iconSet.IconSet = ExcelIconSetType.ThreeSymbols
iconSet.IconCriteria(1).Type = ConditionValueType.Percent
iconSet.IconCriteria(1).Value = "50"
iconSet.IconCriteria(2).Type = ConditionValueType.Percent
iconSet.IconCriteria(2).Value = "50"
iconSet.ShowIconOnly = True
Dim fileName As String = "ConditionalFormatting.xlsx"
workbook.SaveAs(fileName)
End Using

```


UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create data bars for the data in specified range
    IConditionalFormats conditionalFormats =
    worksheet.Range["C7:C46"].ConditionalFormats;
    IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
    conditionalFormat.FormatType = ExcelCFTType.DataBar;
    IDataBar dataBar = conditionalFormat.DataBar;
    //Set the constraints
    dataBar.MinPoint.Type = ConditionValueType.LowestValue;
    dataBar.MaxPoint.Type = ConditionValueType.HighestValue;
    //Set color for Bar
    dataBar.BarColor = Color.FromArgb(255, 156, 208, 243);
    //Hide the values in data bar
    dataBar.ShowValue = false;
    dataBar.BarColor = Color.FromArgb(255, 0, 255, 255);
    //Create color scales for the data in specified range
    conditionalFormats = worksheet.Range["D7:D46"].ConditionalFormats;
    conditionalFormat = conditionalFormats.AddCondition();
    conditionalFormat.FormatType = ExcelCFTType.ColorScale;
    IColorScale colorScale = conditionalFormat.ColorScale;
    //Sets 3 - color scale
    colorScale.SetConditionCount(3);
    colorScale.Criteria[0].FormatColorRGB = Color.FromArgb(255, 230, 197, 218);
    colorScale.Criteria[0].Type = ConditionValueType.LowestValue;
    colorScale.Criteria[0].Value = "0";
    colorScale.Criteria[1].FormatColorRGB = Color.FromArgb(255, 244, 210, 178);
    colorScale.Criteria[1].Type = ConditionValueType.Percentile;
    colorScale.Criteria[1].Value = "50";
    colorScale.Criteria[2].FormatColorRGB = Color.FromArgb(255, 245, 247, 171);
    colorScale.Criteria[2].Type = ConditionValueType.HighestValue;
    colorScale.Criteria[2].Value = "0";
    conditionalFormat.FirstFormulaR1C1 = "=R[1]C[0]";
    conditionalFormat.SecondFormulaR1C1 = "=R[1]C[1]";
    //Create icon sets for the data in specified range
    conditionalFormats = worksheet.Range["E7:E46"].ConditionalFormats;
    conditionalFormat = conditionalFormats.AddCondition();
    conditionalFormat.FormatType = ExcelCFTType.IconSet;
    IIconSet iconSet = conditionalFormat.IconSet;
    //Apply three symbols icon and hide the data in the specified range
    iconSet.IconSet = ExcelIconSetType.ThreeSymbols;
    iconSet.IconCriteria[1].Type = ConditionValueType.Percent;
    iconSet.IconCriteria[1].Value = "50";
}

```

```

iconSet.IconCriteria[2].Type = ConditionValueType.Percent;
iconSet.IconCriteria[2].Value = "50";
iconSet.ShowIconOnly = true;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ConditionalFormatting";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("SampleTemplate.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream,
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create data bars for the data in specified range
    IConditionalFormats conditionalFormats =
    worksheet.Range["C7:C46"].ConditionalFormats;
    IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
    conditionalFormat.FormatType = ExcelCFType.DataBar;
    IDataBar dataBar = conditionalFormat.DataBar;
    //Set the constraints
    dataBar.MinPoint.Type = ConditionValueType.LowestValue;
    dataBar.MaxPoint.Type = ConditionValueType.HighestValue;
    //Set color for Bar
    dataBar.BarColor = Color.FromArgb(156, 208, 243);
    //Hide the values in data bar
    dataBar.ShowValue = false;
    dataBar.BarColor = Color.Aqua;
    //Create color scales for the data in specified range
    conditionalFormats = worksheet.Range["D7:D46"].ConditionalFormats;
    conditionalFormat = conditionalFormats.AddCondition();
    conditionalFormat.FormatType = ExcelCFType.ColorScale;
    IColorScale colorScale = conditionalFormat.ColorScale;
    //Sets 3 - color scale
    colorScale.SetConditionCount(3);
    colorScale.Criteria[0].FormatColorRGB = Color.FromArgb(230, 197, 218);
    colorScale.Criteria[0].Type = ConditionValueType.LowestValue;
    colorScale.Criteria[0].Value = "0";
    colorScale.Criteria[1].FormatColorRGB = Color.FromArgb(244, 210, 178);
    colorScale.Criteria[1].Type = ConditionValueType.Percentile;
    colorScale.Criteria[1].Value = "50";
    colorScale.Criteria[2].FormatColorRGB = Color.FromArgb(245, 247, 171);
    colorScale.Criteria[2].Type = ConditionValueType.HighestValue;
    colorScale.Criteria[2].Value = "0";
}

```

```

conditionalFormat.FirstFormulaR1C1 = "=R[1]C[0]";
conditionalFormat.SecondFormulaR1C1 = "=R[1]C[1]";
//Create icon sets for the data in specified range
conditionalFormats = worksheet.Range["E7:E46"].ConditionalFormats;
conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCFTType.IconSet;
IIconSet iconSet = conditionalFormat.IconSet;
//Apply three symbols icon and hide the data in the specified range
iconSet.IconSet = ExcelIconSetType.ThreeSymbols;
iconSet.IconCriteria[1].Type = ConditionValueType.Percent;
iconSet.IconCriteria[1].Value = "50";
iconSet.IconCriteria[2].Type = ConditionValueType.Percent;
iconSet.IconCriteria[2].Value = "50";
iconSet.ShowIconOnly = true;
//Saving the workbook as stream
FileStream stream = new FileStream("ConditionalFormatting.xlsx",
FileMode.Create, FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

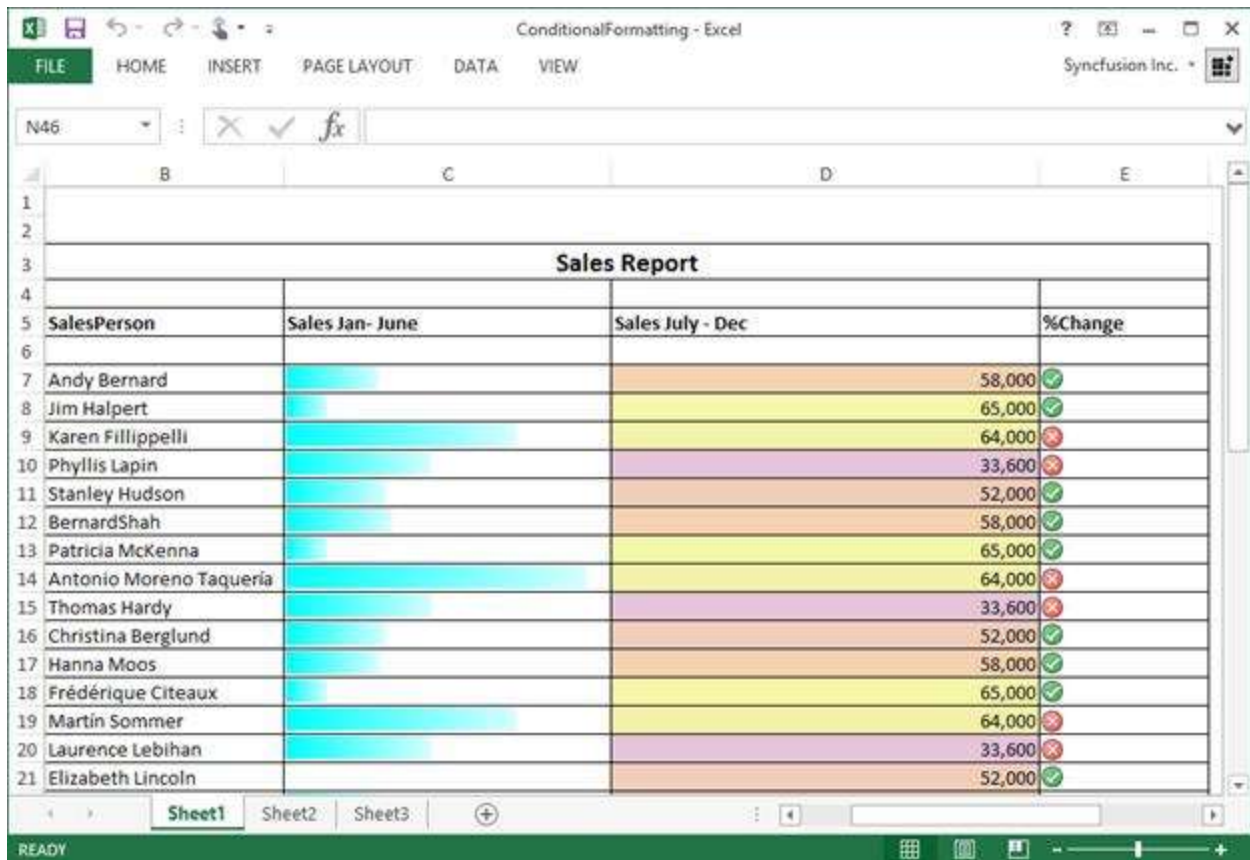
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    /"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.SampleTemplate.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create data bars for the data in specified range
    IConditionalFormats conditionalFormats =
    worksheet.Range["C7:C46"].ConditionalFormats;
    IConditionalFormat conditionalFormat = conditionalFormats.AddCondition();
    conditionalFormat.FormatType = ExcelCFTType.DataBar;
    IDataBar dataBar = conditionalFormat.DataBar;
    //Set the constraints
    dataBar.MinPoint.Type = ConditionValueType.LowestValue;
    dataBar.MaxPoint.Type = ConditionValueType.HighestValue;
    //Set color for Bar
    dataBar.BarColor = Syncfusion.Drawing.Color.FromArgb(156, 208, 243);
    //Hide the values in data bar
    dataBar.ShowValue = false;
    dataBar.BarColor = Syncfusion.Drawing.Color.Aqua;
    //Create color scales for the data in specified range
    conditionalFormats = worksheet.Range["D7:D46"].ConditionalFormats;
    conditionalFormat = conditionalFormats.AddCondition();
    conditionalFormat.FormatType = ExcelCFTType.ColorScale;
    IColorScale colorScale = conditionalFormat.ColorScale;
    //Sets 3 - color scale
    colorScale.SetConditionCount(3);
}

```

```

colorScale.Criteria[0].FormatColorRGB =
Syncfusion.Drawing.Color.FromArgb(230, 197, 218);
colorScale.Criteria[0].Type = ConditionValueType.LowestValue;
colorScale.Criteria[0].Value = "0";
colorScale.Criteria[1].FormatColorRGB =
Syncfusion.Drawing.Color.FromArgb(244, 210, 178);
colorScale.Criteria[1].Type = ConditionValueType.Percentile;
colorScale.Criteria[1].Value = "50";
colorScale.Criteria[2].FormatColorRGB =
Syncfusion.Drawing.Color.FromArgb(245, 247, 171);
colorScale.Criteria[2].Type = ConditionValueType.HighestValue;
colorScale.Criteria[2].Value = "0";
conditionalFormat.FirstFormulaR1C1 = "=R[1]C[0]";
conditionalFormat.SecondFormulaR1C1 = "=R[1]C[1]";
//Create icon sets for the data in specified range
conditionalFormats = worksheet.Range["E7:E46"].ConditionalFormats;
conditionalFormat = conditionalFormats.AddCondition();
conditionalFormat.FormatType = ExcelCFTType.IconSet;
IIconSet iconSet = conditionalFormat.IconSet;
//Apply three symbols icon and hide the data in the specified range
iconSet.IconSet = ExcelIconSetType.ThreeSymbols;
iconSet.IconCriteria[1].Type = ConditionValueType.Percent;
iconSet.IconCriteria[1].Value = "50";
iconSet.IconCriteria[2].Type = ConditionValueType.Percent;
iconSet.IconCriteria[2].Value = "50";
iconSet.ShowIconOnly = true;
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Condit
ionalFormatting.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ConditionalFormatt
ing.xlsx", "application/msexcel", stream);
}
}

```



Excel with Conditional Formatting

Note: XlsIO visualization has been enhanced with backward compatibility for Advanced Conditional Formatting.

Working with Data Validation

Data Validation is a list of rules to the data that can be entered in a cell. This can be applied by using **IDataValidation** interface. XlsIO supports following validation types.

- Text Length Validation
- Time Validation
- List Validation
- Number Validation
- Date Validation
- Custom Validation

Text Length Validation

The following code snippet illustrates how to set text length validation.

C#

```
//Data validation for text length
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.TextLength;
//Text length should be lesser than 5 characters
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
```

```
validation.FirstFormula = "0";
validation.SecondFormula = "5";
```

VB.NET

```
'Data validation for text length
Dim validation As IDataValidation = sheet.Range("A3").DataValidation
validation.AllowType = ExcelDataType.TextLength
'Text length should be lesser than 5 characters
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between
validation.FirstFormula = "0"
validation.SecondFormula = "5"
```

UWP

```
//Data validation for text length
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.TextLength;
//Text length should be lesser than 5 characters
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstFormula = "0";
validation.SecondFormula = "5";
```

ASP.NET CORE

```
//Data validation for text length
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.TextLength;
//Text length should be lesser than 5 characters
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstFormula = "0";
validation.SecondFormula = "5";
```

XAMARIN

```
//Data validation for text length
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.TextLength;
//Text length should be lesser than 5 characters
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstFormula = "0";
validation.SecondFormula = "5";
```

Time Validation

The following code snippet illustrates how to set time validation.

C#

```
//Data validation for time
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.Time;
//Time between 10:00 and 12:00 'o Clock
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstFormula = "10.00";
```

```
validation.SecondFormula = "12.00";
```

VB.NET

```
'Data validation for time
Dim validation As IDataValidation = sheet.Range("A3").DataValidation
validation.AllowType = ExcelDataType.Time
'Time between 10:00 and 12:00 'o Clock
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between
validation.FirstFormula = "10.00"
validation.SecondFormula = "12.00"
```

UWP

```
//Data validation for time
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.Time;
//Time between 10:00 and 12:00 'o Clock
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstFormula = "10.00";
validation.SecondFormula = "12.00";
```

ASP.NET CORE

```
//Data validation for time
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.Time;
//Time between 10:00 and 12:00 'o Clock
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstFormula = "10.00";
validation.SecondFormula = "12.00";
```

XAMARIN

```
//Data validation for time
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.Time;
//Time between 10:00 and 12:00 'o Clock
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstFormula = "10.00";
validation.SecondFormula = "12.00";
```

List Validation

The following code snippet illustrates how to set list validation.

C#

```
//Data validation for list
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.ListOfValues = new string[] { "ListItem1", "ListItem2",
"ListItem3" };;
```

VB.NET

```
'Data validation for list  
Dim validation As IDataValidation = sheet.Range("A3").DataValidation  
validation.ListOfValues = New String() { "ListItem1", "ListItem2",  
"ListItem3" }
```

UWP

```
//Data validation for list  
IDataValidation validation = sheet.Range["A3"].DataValidation;  
validation.ListOfValues = new string[] { "ListItem1", "ListItem2",  
"ListItem3" };
```

ASP.NET CORE

```
//Data validation for list  
IDataValidation validation = sheet.Range["A3"].DataValidation;  
validation.ListOfValues = new string[] { "ListItem1", "ListItem2",  
"ListItem3" };
```

XAMARIN

```
//Data validation for list  
IDataValidation validation = sheet.Range["A3"].DataValidation;  
validation.ListOfValues = new string[] { "ListItem1", "ListItem2",  
"ListItem3" };
```

Note: The ListOfValues property should be used when the values in the Data Validation list are entered manually whose limit is only 255 characters including separators.

Number Validation

The following code snippet illustrates how to set number validation.

C#

```
//Data validation for number  
IDataValidation validation = sheet.Range["A3"].DataValidation;  
validation.AllowType = ExcelDataType.Integer;  
//Value between 0 to 10  
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;  
validation.FirstFormula = "0";  
validation.SecondFormula = "10";
```

VB.NET

```
'Data validation for number  
Dim validation As IDataValidation = sheet.Range("A3").DataValidation  
validation.AllowType = ExcelDataType.Integer  
'Value between 0 to 10  
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between  
validation.FirstFormula = "0"  
validation.SecondFormula = "10"
```

UWP


```
//Data validation for number
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.Integer;
//Value between 0 to 10
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstFormula = "0";
validation.SecondFormula = "10";
```

ASP.NET CORE

```
//Data validation for number
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.Integer;
//Value between 0 to 10
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstFormula = "0";
validation.SecondFormula = "10";
```

XAMARIN

```
//Data validation for number
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.Integer;
//Value between 0 to 10
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstFormula = "0";
validation.SecondFormula = "10";
```

Date Validation

The following code snippet illustrates how to set date validation.

C#

```
//Data validation for date
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.Date;
//Date between 10/5/2003 to 10/5/2004
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstDateTime = new DateTime(2003, 5, 10);
validation.SecondDateTime = new DateTime(2004, 5, 10);
```

VB.NET

```
'Data validation for date
Dim validation As IDataValidation = sheet.Range("A3").DataValidation
validation.AllowType = ExcelDataType.Date
'Date between 10/5/2003 to 10/5/2004
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between
validation.FirstDateTime = New DateTime(2003, 5, 10)
validation.SecondDateTime = New DateTime(2004, 5, 10)
```

UWP

```
//Data validation for date
```

```

IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.Date;
//Date between 10/5/2003 to 10/5/2004
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstDateTime = new DateTime(2003, 5, 10);
validation.SecondDateTime = new DateTime(2004, 5, 10);

```

ASP.NET CORE

```

//Data validation for date
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.Date;
//Date between 10/5/2003 to 10/5/2004
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstDateTime = new DateTime(2003, 5, 10);
validation.SecondDateTime = new DateTime(2004, 5, 10);

```

XAMARIN

```

//Data validation for date
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.Date;
//Date between 10/5/2003 to 10/5/2004
validation.CompareOperator = ExcelDataValidationComparisonOperator.Between;
validation.FirstDateTime = new DateTime(2003, 5, 10);
validation.SecondDateTime = new DateTime(2004, 5, 10);

```

Custom Validation

Custom validation can be set to a cell with its **AllowType** as **User**. The following code snippet illustrates how to set custom validation.

C#

```

//Data validation for custom data
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.User;
validation.FirstFormula = "=A1>10";

```

VB.NET

```

'Data validation for custom data
Dim validation As IDataValidation = sheet.Range("A3").DataValidation
validation.AllowType = ExcelDataType.User
validation.FirstFormula = "=A1>10"

```

UWP

```

//Data validation for custom data
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.User;
validation.FirstFormula = "=A1>10";

```

ASP.NET CORE

```
//Data validation for custom data
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.User;
validation.FirstFormula = "=A1>10";
```

XAMARIN

```
//Data validation for custom data
IDataValidation validation = sheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.User;
validation.FirstFormula = "=A1>10";
```

The following code snippet shows all the data validation supports discussed previously.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];

    //Data Validation for Text Length
    IDataValidation txtLengthValidation = worksheet.Range["A3"].DataValidation;
    worksheet.Range["A1"].Text = "Enter the Text in A3";
    worksheet.Range["A1"].AutofitColumns();
    txtLengthValidation.AllowType = ExcelDataType.TextLength;
    txtLengthValidation.CompareOperator =
        ExcelDataValidationComparisonOperator.Between;
    txtLengthValidation.FirstFormula = "0";
    txtLengthValidation.SecondFormula = "5";
    //Shows the error message
    txtLengthValidation.ShowErrorBox = true;
    txtLengthValidation.ErrorBoxText = "Text length should be lesser than 5 characters";
    txtLengthValidation.ErrorBoxTitle = "ERROR";
    txtLengthValidation.PromptBoxText = "Data validation for text length";
    txtLengthValidation.ShowPromptBox = true;

    //Data Validation for Time
    IDataValidation timeValidation = worksheet.Range["B3"].DataValidation;
    worksheet.Range["B1"].Text = "Enter the time between 10:00 and 12:00 'o Clock in B3";
    worksheet.Range["B1"].AutofitColumns();
    timeValidation.AllowType = ExcelDataType.Time;
    timeValidation.CompareOperator =
        ExcelDataValidationComparisonOperator.Between;
    timeValidation.FirstFormula = "10.00";
    timeValidation.SecondFormula = "12.00";
    //Shows the error message
    timeValidation.ShowErrorBox = true;
    timeValidation.ErrorBoxText = "Enter a correct time";
    timeValidation.ErrorBoxTitle = "ERROR";
    timeValidation.PromptBoxText = "Data validation for time";
    timeValidation.ShowPromptBox = true;

    //Data Validation for List
    IDataValidation listValidation = worksheet.Range["C3"].DataValidation;
```

```

worksheet.Range["C1"].Text = "Data Validation List in C3";
worksheet.Range["C1"].AutofitColumns();
listValidation.ListOfValues = new string[] { "ListItem1", "ListItem2",
"ListItem3" };
//Shows the error message
listValidation.ErrorBoxText = "Choose the value from the list";
listValidation.ErrorBoxTitle = "ERROR";
listValidation.PromptBoxText = "Data validation for list";
listValidation.IsPromptBoxVisible = true;
listValidation.ShowPromptBox = true;
//Data Validation for Numbers
IDataValidation numberValidation = worksheet.Range["D3"].DataValidation;
worksheet.Range["D1"].Text = "Enter the Number in D3";
worksheet.Range["D1"].AutofitColumns();
numberValidation.AllowType = ExcelDataType.Integer;
numberValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between;
numberValidation.FirstFormula = "0";
numberValidation.SecondFormula = "10";
//Shows the error message
numberValidation.ShowErrorBox = true;
numberValidation.ErrorBoxText = "Enter Value between 0 to 10";
numberValidation.ErrorBoxTitle = "ERROR";
numberValidation.PromptBoxText = "Data validation for numbers";
numberValidation.ShowPromptBox = true;
//Data Validation for Date
IDataValidation dateValidation = worksheet.Range["E3"].DataValidation;
worksheet.Range["E1"].Text = "Enter the Date in E3";
worksheet.Range["E1"].AutofitColumns();
dateValidation.AllowType = ExcelDataType.Date;
dateValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between;
dateValidation.FirstDateTime = new DateTime(2003, 5, 10);
dateValidation.SecondDateTime = new DateTime(2004, 5, 10);
//Shows the error message
dateValidation.ShowErrorBox = true;
dateValidation.ErrorBoxText = "Enter Value between 10/5/2003 to 10/5/2004";
dateValidation.ErrorBoxTitle = "ERROR";
dateValidation.PromptBoxText = "Data validation for date";
dateValidation.ShowPromptBox = true;
//Data validation for custom data
IDataValidation validation = worksheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.User;
validation.FirstFormula = "=A1>10";
//Shows the error message
validation.ErrorBoxText = "Enter value in A1 greater than 10";
validation.ErrorBoxTitle = "ERROR";
validation.PromptBoxText = "Custom DataValidation";
validation.ShowPromptBox = true;
workbook.SaveAs("DataValidation.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel

```

```

application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Data Validation for Text Length
Dim txtLengthValidation As IDataValidation =
worksheet.Range("A3").DataValidation
worksheet.Range("A1").Text = "Enter the Text in A3"
worksheet.Range("A1").AutofitColumns()
txtLengthValidation.AllowType = ExcelDataType.TextLength
txtLengthValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between
txtLengthValidation.FirstFormula = "0"
txtLengthValidation.SecondFormula = "5"
'Shows the error message
txtLengthValidation.ShowErrorBox = True
txtLengthValidation.ErrorBoxText = "Text length should be lesser than 5
characters"
txtLengthValidation.ErrorBoxTitle = "ERROR"
txtLengthValidation.PromptBoxText = "Data validation for text length"
txtLengthValidation.ShowPromptBox = True
'Data Validation for Time
Dim timeValidation As IDataValidation = worksheet.Range("B3").DataValidation
worksheet.Range("B1").Text = "Enter the time between 10:00 and 12:00 'o
Clock in B3"
worksheet.Range("B1").AutofitColumns()
timeValidation.AllowType = ExcelDataType.Time
timeValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between
timeValidation.FirstFormula = "10.00"
timeValidation.SecondFormula = "12.00"
'Shows the error message
timeValidation.ShowErrorBox = True
timeValidation.ErrorBoxText = "Enter a correct time"
timeValidation.ErrorBoxTitle = "ERROR"
timeValidation.PromptBoxText = "Data validation for time"
timeValidation.ShowPromptBox = True
'Data Validation for List
Dim listValidation As IDataValidation = worksheet.Range("C3").DataValidation
worksheet.Range("C1").Text = "Data Validation List in C3"
worksheet.Range("C1").AutofitColumns()
listValidation.ListOfValues = New String() {"ListItem1", "ListItem2",
"ListItem3"}
'Shows the error message
listValidation.ErrorBoxText = "Choose the value from the list"
listValidation.ErrorBoxTitle = "ERROR"
listValidation.PromptBoxText = "Data validation for list"
listValidation.IsPromptBoxVisible = True
listValidation.ShowPromptBox = True
'Data Validation for Numbers
Dim numberValidation As IDataValidation =
worksheet.Range("D3").DataValidation
worksheet.Range("D1").Text = "Enter the Number in D3"
worksheet.Range("D1").AutofitColumns()
numberValidation.AllowType = ExcelDataType.Integer
numberValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between
numberValidation.FirstFormula = "0"

```

```

numberValidation.SecondFormula = "10"
'Shows the error message
numberValidation.ShowErrorBox = True
numberValidation.ErrorBoxText = "Enter Value between 0 to 10"
numberValidation.ErrorBoxTitle = "ERROR"
numberValidation.PromptBoxText = "Data validation for numbers"
numberValidation.ShowPromptBox = True
'Data Validation for Date
Dim dateValidation As IDataValidation = worksheet.Range("E3").DataValidation
worksheet.Range("E1").Text = "Enter the Date in E3"
worksheet.Range("E1").AutofitColumns()
dateValidation.AllowType = ExcelDataType.Date
dateValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between
dateValidation.FirstDateTime = New DateTime(2003, 5, 10)
dateValidation.SecondDateTime = New DateTime(2004, 5, 10)
'Shows the error message
dateValidation.ShowErrorBox = True
dateValidation.ErrorBoxText = "Enter Value between 10/5/2003 to 10/5/2004"
dateValidation.ErrorBoxTitle = "ERROR"
dateValidation.PromptBoxText = "Data validation for date"
dateValidation.ShowPromptBox = True
'Data validation for custom data
Dim validation As IDataValidation = worksheet.Range("A3").DataValidation
validation.AllowType = ExcelDataType.User
validation.FirstFormula = "=A1>10"
'Shows the error message
validation.ErrorBoxText = "Enter value in A1 greater than 10"
validation.ErrorBoxTitle = "ERROR"
validation.PromptBoxText = "Custom DataValidation"
validation.ShowPromptBox = True
workbook.SaveAs("DataValidation.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Data Validation for Text Length
    IDataValidation txtLengthValidation = worksheet.Range["A3"].DataValidation;
    worksheet.Range["A1"].Text = "Enter the Text in A3";
    worksheet.Range["A1"].AutofitColumns();
    txtLengthValidation.AllowType = ExcelDataType.TextLength;
    txtLengthValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between;
    txtLengthValidation.FirstFormula = "0";
    txtLengthValidation.SecondFormula = "5";
    //Shows the error message
    txtLengthValidation.ShowErrorBox = true;
    txtLengthValidation.ErrorBoxText = "Text length should be lesser than 5
characters";
    txtLengthValidation.ErrorBoxTitle = "ERROR";
}

```

```
txtLengthValidation.PromptBoxText = "Data validation for text length";
txtLengthValidation.ShowPromptBox = true;
//Data Validation for Time
IDataValidation timeValidation = worksheet.Range["B3"].DataValidation;
worksheet.Range["B1"].Text = "Enter the time between 10:00 and 12:00 'o
Clock in B3";
worksheet.Range["B1"].AutofitColumns();
timeValidation.AllowType = ExcelDataType.Time;
timeValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between;
timeValidation.FirstFormula = "10.00";
timeValidation.SecondFormula = "12.00";
//Shows the error message
timeValidation.ShowErrorBox = true;
timeValidation.ErrorBoxText = "Enter a correct time";
timeValidation.ErrorBoxTitle = "ERROR";
timeValidation.PromptBoxText = "Data validation for time";
timeValidation.ShowPromptBox = true;
//Data Validation for List
IDataValidation listValidation = worksheet.Range["C3"].DataValidation;
worksheet.Range["C1"].Text = "Data Validation List in C3";
worksheet.Range["C1"].AutofitColumns();
listValidation.ListOfValues = new string[] { "ListItem1", "ListItem2",
"ListItem3" };
//Shows the error message
listValidation.ErrorBoxText = "Choose the value from the list";
listValidation.ErrorBoxTitle = "ERROR";
listValidation.PromptBoxText = "Data validation for list";
listValidation.IsPromptBoxVisible = true;
listValidation.ShowPromptBox = true;
//Data Validation for Numbers
IDataValidation numberValidation = worksheet.Range["D3"].DataValidation;
worksheet.Range["D1"].Text = "Enter the Number in D3";
worksheet.Range["D1"].AutofitColumns();
numberValidation.AllowType = ExcelDataType.Integer;
numberValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between;
numberValidation.FirstFormula = "0";
numberValidation.SecondFormula = "10";
//Shows the error message
numberValidation.ShowErrorBox = true;
numberValidation.ErrorBoxText = "Enter Value between 0 to 10";
numberValidation.ErrorBoxTitle = "ERROR";
numberValidation.PromptBoxText = "Data validation for numbers";
numberValidation.ShowPromptBox = true;
//Data Validation for Date
IDataValidation dateValidation = worksheet.Range["E3"].DataValidation;
worksheet.Range["E1"].Text = "Enter the Date in E3";
worksheet.Range["E1"].AutofitColumns();
dateValidation.AllowType = ExcelDataType.Date;
dateValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between;
dateValidation.FirstDateTime = new DateTime(2003, 5, 10);
dateValidation.SecondDateTime = new DateTime(2004, 5, 10);
//Shows the error message
dateValidation.ShowErrorBox = true;
dateValidation.ErrorBoxText = "Enter Value between 10/5/2003 to 10/5/2004";
```

```

dateValidation.ErrorBoxTitle = "ERROR";
dateValidation.PromptBoxText = "Data validation for date";
dateValidation.ShowPromptBox = true;
//Data validation for custom data
IDataValidation validation = worksheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.User;
validation.FirstFormula = "=A1>10";
//Shows the error message
validation.ErrorBoxText = "Enter value in A1 greater than 10";
validation.ErrorBoxTitle = "ERROR";
validation.PromptBoxText = "Custom DataValidation";
validation.ShowPromptBox = true;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "DataValidation";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Data Validation for Text Length
    IDataValidation txtLengthValidation = worksheet.Range["A3"].DataValidation;
    worksheet.Range["A1"].Text = "Enter the Text in A3";
    worksheet.Range["A1"].AutofitColumns();
    txtLengthValidation.AllowType = ExcelDataType.TextLength;
    txtLengthValidation.CompareOperator =
    ExcelDataValidationComparisonOperator.Between;
    txtLengthValidation.FirstFormula = "0";
    txtLengthValidation.SecondFormula = "5";
    //Shows the error message
    txtLengthValidation.ShowErrorBox = true;
    txtLengthValidation.ErrorBoxText = "Text length should be lesser than 5
characters";
    txtLengthValidation.ErrorBoxTitle = "ERROR";
    txtLengthValidation.PromptBoxText = "Data validation for text length";
    txtLengthValidation.ShowPromptBox = true;
    //Data Validation for Time
    IDataValidation timeValidation = worksheet.Range["B3"].DataValidation;
    worksheet.Range["B1"].Text = "Enter the time between 10:00 and 12:00 'o
Clock in B3";
    worksheet.Range["B1"].AutofitColumns();
    timeValidation.AllowType = ExcelDataType.Time;
    timeValidation.CompareOperator =
    ExcelDataValidationComparisonOperator.Between;
}

```



```

timeValidation.FirstFormula = "10.00";
timeValidation.SecondFormula = "12.00";
//Shows the error message
timeValidation.ShowErrorBox = true;
timeValidation.ErrorBoxText = "Enter a correct time";
timeValidation.ErrorBoxTitle = "ERROR";
timeValidation.PromptBoxText = "Data validation for time";
timeValidation.ShowPromptBox = true;
//Data Validation for List
IDataValidation listValidation = worksheet.Range["C3"].DataValidation;
worksheet.Range["C1"].Text = "Data Validation List in C3";
worksheet.Range["C1"].AutofitColumns();
listValidation.ListOfValues = new string[] { "ListItem1", "ListItem2",
"ListItem3" };
//Shows the error message
listValidation.ErrorBoxText = "Choose the value from the list";
listValidation.ErrorBoxTitle = "ERROR";
listValidation.PromptBoxText = "Data validation for list";
listValidation.IsPromptBoxVisible = true;
listValidation.ShowPromptBox = true;
//Data Validation for Numbers
IDataValidation numberValidation = worksheet.Range["D3"].DataValidation;
worksheet.Range["D1"].Text = "Enter the Number in D3";
worksheet.Range["D1"].AutofitColumns();
numberValidation.AllowType = ExcelDataType.Integer;
numberValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between;
numberValidation.FirstFormula = "0";
numberValidation.SecondFormula = "10";
//Shows the error message
numberValidation.ShowErrorBox = true;
numberValidation.ErrorBoxText = "Enter Value between 0 to 10";
numberValidation.ErrorBoxTitle = "ERROR";
numberValidation.PromptBoxText = "Data validation for numbers";
numberValidation.ShowPromptBox = true;
//Data Validation for Date
IDataValidation dateValidation = worksheet.Range["E3"].DataValidation;
worksheet.Range["E1"].Text = "Enter the Date in E3";
worksheet.Range["E1"].AutofitColumns();
dateValidation.AllowType = ExcelDataType.Date;
dateValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between;
dateValidation.FirstDateTime = new DateTime(2003, 5, 10);
dateValidation.SecondDateTime = new DateTime(2004, 5, 10);
//Shows the error message
dateValidation.ShowErrorBox = true;
dateValidation.ErrorBoxText = "Enter Value between 10/5/2003 to 10/5/2004";
dateValidation.ErrorBoxTitle = "ERROR";
dateValidation.PromptBoxText = "Data validation for date";
dateValidation.ShowPromptBox = true;
//Data validation for custom data
IDataValidation validation = worksheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.User;
validation.FirstFormula = "=A1>10";
//Shows the error message
validation.ErrorBoxText = "Enter value in A1 greater than 10";
validation.ErrorBoxTitle = "ERROR";

```

```
validation.PromptBoxText = "Custom DataValidation";
validation.ShowPromptBox = true;
FileStream file = new FileStream("DataValidation.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(file);
file.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];

    //Data Validation for Text Length
    IDataValidation txtLengthValidation = worksheet.Range["A3"].DataValidation;
    worksheet.Range["A1"].Text = "Enter the Text in A3";
    worksheet.Range["A1"].AutofitColumns();
    txtLengthValidation.AllowType = ExcelDataType.TextLength;
    txtLengthValidation.CompareOperator =
        ExcelDataValidationComparisonOperator.Between;
    txtLengthValidation.FirstFormula = "0";
    txtLengthValidation.SecondFormula = "5";
    //Shows the error message
    txtLengthValidation.ShowErrorBox = true;
    txtLengthValidation.ErrorBoxText = "Text length should be lesser than 5
    characters";
    txtLengthValidation.ErrorBoxTitle = "ERROR";
    txtLengthValidation.PromptBoxText = "Data validation for text length";
    txtLengthValidation.ShowPromptBox = true;

    //Data Validation for Time
    IDataValidation timeValidation = worksheet.Range["B3"].DataValidation;
    worksheet.Range["B1"].Text = "Enter the time between 10:00 and 12:00 'o
    Clock in B3";
    worksheet.Range["B1"].AutofitColumns();
    timeValidation.AllowType = ExcelDataType.Time;
    timeValidation.CompareOperator =
        ExcelDataValidationComparisonOperator.Between;
    timeValidation.FirstFormula = "10.00";
    timeValidation.SecondFormula = "12.00";
    //Shows the error message
    timeValidation.ShowErrorBox = true;
    timeValidation.ErrorBoxText = "Enter a correct time";
    timeValidation.ErrorBoxTitle = "ERROR";
    timeValidation.PromptBoxText = "Data validation for time";
    timeValidation.ShowPromptBox = true;

    //Data Validation for List
    IDataValidation listValidation = worksheet.Range["C3"].DataValidation;
    worksheet.Range["C1"].Text = "Data Validation List in C3";
    worksheet.Range["C1"].AutofitColumns();
    listValidation.ListOfValues = new string[] { "ListItem1", "ListItem2",
    "ListItem3" };
    //Shows the error message
    listValidation.ErrorBoxText = "Choose the value from the list";
}
```

```

listValidation.ErrorBoxTitle = "ERROR";
listValidation.PromptBoxText = "Data validation for list";
listValidation.IsPromptBoxVisible = true;
listValidation.ShowPromptBox = true;
//Data Validation for Numbers
IDataValidation numberValidation = worksheet.Range["D3"].DataValidation;
worksheet.Range["D1"].Text = "Enter the Number in D3";
worksheet.Range["D1"].AutofitColumns();
numberValidation.AllowType = ExcelDataType.Integer;
numberValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between;
numberValidation.FirstFormula = "0";
numberValidation.SecondFormula = "10";
//Shows the error message
numberValidation.ShowErrorBox = true;
numberValidation.ErrorBoxText = "Enter Value between 0 to 10";
numberValidation.ErrorBoxTitle = "ERROR";
numberValidation.PromptBoxText = "Data validation for numbers";
numberValidation.ShowPromptBox = true;
//Data Validation for Date
IDataValidation dateValidation = worksheet.Range["E3"].DataValidation;
worksheet.Range["E1"].Text = "Enter the Date in E3";
worksheet.Range["E1"].AutofitColumns();
dateValidation.AllowType = ExcelDataType.Date;
dateValidation.CompareOperator =
ExcelDataValidationComparisonOperator.Between;
dateValidation.FirstDateTime = new DateTime(2003, 5, 10);
dateValidation.SecondDateTime = new DateTime(2004, 5, 10);
//Shows the error message
dateValidation.ShowErrorBox = true;
dateValidation.ErrorBoxText = "Enter Value between 10/5/2003 to 10/5/2004";
dateValidation.ErrorBoxTitle = "ERROR";
dateValidation.PromptBoxText = "Data validation for date";
dateValidation.ShowPromptBox = true;
//Data validation for custom data
IDataValidation validation = worksheet.Range["A3"].DataValidation;
validation.AllowType = ExcelDataType.User;
validation.FirstFormula = "=A1>10";
//Shows the error message
validation.ErrorBoxText = "Enter value in A1 greater than 10";
validation.ErrorBoxTitle = "ERROR";
validation.PromptBoxText = "Custom DataValidation";
validation.ShowPromptBox = true;
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("DataVa
lidation.xlsx", "application/msexcel", stream);
}

```

```
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("DataValidation.xlsx", "application/msexcel", stream);
}
}
```

Working with Charts

Essential XlsIO has support for creating and modifying Excel charts inside a workbook or as a [chart worksheet](#).

Creating a Chart

The **IShape** interface represents the chart in a worksheet. A chart can be created either through the existing data in the worksheet, directly entering series or by adding series one by one.

The following code example illustrates how to create a chart through the existing data in the worksheet.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a Chart
    IChartShape chart = sheet.Charts.Add();
    //Set Chart Type
    chart.ChartType = ExcelChartType.Column_Clustered;
    //Set data range in the worksheet
    chart.DataRange = sheet.Range["A1:E5"];
    workbook.SaveAs("Chart.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
    ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Create a Chart
Dim chart As IChartShape = sheet.Charts.Add()
'Set Chart Type
chart.ChartType = ExcelChartType.Column_Clustered
'Set data range in the worksheet
chart.DataRange = sheet.Range("A1:E5")
workbook.SaveAs("Chart.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
```

```

{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a Chart
    IChartShape chart = sheet.Charts.Add();
    //Set Chart Type
    chart.ChartType = ExcelChartType.Column_Clustered;
    //Set data range in the worksheet
    chart.DataRange = sheet.Range["A1:E5"];
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Chart";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a Chart
    IChartShape chart = sheet.Charts.Add();
    //Set Chart Type
    chart.ChartType = ExcelChartType.Column_Clustered;
    //Set data range in the worksheet
    chart.DataRange = sheet.Range["A1:E5"];
    //Saving the workbook as stream
    FileStream stream = new FileStream("Chart.xlsx", FileMode.Create,
        FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    // "App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    // Create a Chart
    IChartShape chart = sheet.Charts.Add();
    // Set Chart Type
    chart.ChartType = ExcelChartType.Column_Clustered;
    // Set data range in the worksheet
    chart.DataRange = sheet.Range["A1:E5"];
    // Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    // Save the document as file and view the saved document
    // The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Chart.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Chart.xlsx", "application/msexcel", stream);
    }
}

```

Creating a Chart from directly entered Values

A chart in XlsIO can also be created from directly entered values. The Following code snippets illustrate how to create a chart from directly entered values.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    object[] yValues = new object[] { 2000, 1000, 1000 };
    object[] xValues = new object[] { "Total Income", "Expenses", "Profit" };
    // Adding series and values
}

```

```

IChartShape chart = sheet.Charts.Add();
IChartSerie serie = chart.Series.Add(ExcelChartType.Pie);
//Enters the X and Y values directly
serie.EnteredDirectlyValues = yValues;
serie.EnteredDirectlyCategoryLabels = xValues;
workbook.SaveAs("Chart.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim yValues As Object() = New Object() {2000, 1000, 1000}
Dim xValues As Object() = New Object() {"Total Income", "Expenses",
"Profit"}
'Adding series and values
Dim chart As IChartShape = sheet.Charts.Add()
Dim serie As IChartSerie = chart.Series.Add(ExcelChartType.Pie)
'Enters the X and Y values directly
serie.EnteredDirectlyValues = yValues
serie.EnteredDirectlyCategoryLabels = xValues
workbook.SaveAs("Chart.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
object[] yValues = new object[] { 2000, 1000, 1000 };
object[] xValues = new object[] { "Total Income", "Expenses", "Profit" };
//Adding series and values
IChartShape chart = sheet.Charts.Add();
IChartSerie serie = chart.Series.Add(ExcelChartType.Pie);
//Enters the X and Y values directly
serie.EnteredDirectlyValues = yValues;
serie.EnteredDirectlyCategoryLabels = xValues;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Chart";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    object[] yValues = new object[] { 2000, 1000, 1000 };
    object[] xValues = new object[] { "Total Income", "Expenses", "Profit" };
    //Adding series and values
    IChartShape chart = sheet.Charts.Add();
    IChartSerie serie = chart.Series.Add(ExcelChartType.Pie);
    //Enters the X and Y values directly
    serie.EnteredDirectlyValues = yValues;
    serie.EnteredDirectlyCategoryLabels = xValues;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Chart.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    object[] yValues = new object[] { 2000, 1000, 1000 };
    object[] xValues = new object[] { "Total Income", "Expenses", "Profit" };
    //Adding series and values
    IChartShape chart = sheet.Charts.Add();
    IChartSerie serie = chart.Series.Add(ExcelChartType.Pie);
    //Enters the X and Y values directly
    serie.EnteredDirectlyValues = yValues;
    serie.EnteredDirectlyCategoryLabels = xValues;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Chart.
        xlsx", "application/msexcel", stream);
    }
    else
    {

```



```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Chart.xlsx",
"application/msexcel", stream);
}
}
```

Creating a Chart by adding Series

A chart can also be created by adding series one by one. The following code illustrates how to create a chart through series.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Inserts the sample data for the chart
    sheet.Range["A1"].Text = "Month";
    sheet.Range["B1"].Text = "Product A";
    sheet.Range["C1"].Text = "Product B";
    //Months
    sheet.Range["A2"].Text = "Jan";
    sheet.Range["A3"].Text = "Feb";
    sheet.Range["A4"].Text = "Mar";
    sheet.Range["A5"].Text = "Apr";
    sheet.Range["A6"].Text = "May";
    //Create a random Data
    Random r = new Random();
    for (int i = 2; i <= 6; i++)
    {
        for (int j = 2; j <= 3; j++)
        {
            sheet.Range[i, j].Number = r.Next(0, 500);
        }
    }
    IChartShape chart = sheet.Charts.Add();
    //Set chart type
    chart.ChartType = ExcelChartType.Line;
    //Set Chart Title
    chart.ChartTitle = "Product Sales comparison";
    //Set first serie
    IChartSerie productA = chart.Series.Add("ProductA");
    productA.Values = sheet.Range["B2:B6"];
    productA.CategoryLabels = sheet.Range["A2:A6"];
    //Set second serie
    IChartSerie productB = chart.Series.Add("ProductB");
    productB.Values = sheet.Range["C2:C6"];
    productB.CategoryLabels = sheet.Range["A2:A6"];
    workbook.SaveAs("Chart.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
```

```

Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Inserting sample data for the chart
sheet.Range("A1").Text = "Month"
sheet.Range("B1").Text = "Product A"
sheet.Range("C1").Text = "Product B"
'Months
sheet.Range("A2").Text = "Jan"
sheet.Range("A3").Text = "Feb"
sheet.Range("A4").Text = "Mar"
sheet.Range("A5").Text = "Apr"
sheet.Range("A6").Text = "May"
'Create a random data
Dim r As Random = New Random
For i As Integer = 2 To 6
For j As Integer = 2 To 3
sheet.Range(i, j).Number = r.Next(0, 500)
Next j
Next i
Dim chart As IChartShape = sheet.Charts.Add()
'Set chart type
chart.ChartType = ExcelChartType.Line
'Set Chart Title
chart.ChartTitle = "Product Sales comparison"
'Set first serie
Dim productA As IChartSerie = chart.Series.Add("ProductA")
productA.Values = sheet.Range("B2:B6")
productA.CategoryLabels = sheet.Range("A2:A6")
'set second serie
Dim productB As IChartSerie = chart.Series.Add("ProductB")
productB.Values = sheet.Range("C2:C6")
productB.CategoryLabels = sheet.Range("A2:A6")
workbook.SaveAs("Chart.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Inserts the sample data for the chart
sheet.Range["A1"].Text = "Month";
sheet.Range["B1"].Text = "Product A";
sheet.Range["C1"].Text = "Product B";
//Months
sheet.Range["A2"].Text = "Jan";
sheet.Range["A3"].Text = "Feb";
sheet.Range["A4"].Text = "Mar";
sheet.Range["A5"].Text = "Apr";
sheet.Range["A6"].Text = "May";
//Create a random Data

```

```

Random r = new Random();
for (int i = 2; i <= 6; i++)
{
    for (int j = 2; j <= 3; j++)
    {
        sheet.Range[i, j].Number = r.Next(0, 500);
    }
}
IChartShape chart = sheet.Charts.Add();
//Set chart type
chart.ChartType = ExcelChartType.Line;
//Set Chart Title
chart.ChartTitle = "Product Sales comparison";
//Set first serie
IChartSerie productA = chart.Series.Add("ProductA");
productA.Values = sheet.Range["B2:B6"];
productA.CategoryLabels = sheet.Range["A2:A6"];
//Set second serie
IChartSerie productB = chart.Series.Add("ProductB");
productB.Values = sheet.Range["C2:C6"];
productB.CategoryLabels = sheet.Range["A2:A6"];
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Chart";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Inserts the sample data for the chart
    sheet.Range["A1"].Text = "Month";
    sheet.Range["B1"].Text = "Product A";
    sheet.Range["C1"].Text = "Product B";
    //Months
    sheet.Range["A2"].Text = "Jan";
    sheet.Range["A3"].Text = "Feb";
    sheet.Range["A4"].Text = "Mar";
    sheet.Range["A5"].Text = "Apr";
    sheet.Range["A6"].Text = "May";
    //Create a random Data
    Random r = new Random();
    for (int i = 2; i <= 6; i++)
    {
        for (int j = 2; j <= 3; j++)

```

```

{
    sheet.Range[i, j].Number = r.Next(0, 500);
}
}
IChartShape chart = sheet.Charts.Add();
//Set chart type
chart.ChartType = ExcelChartType.Line;
//Set Chart Title
chart.ChartTitle = "Product Sales comparison";
//Set first serie
IChartSerie productA = chart.Series.Add("ProductA");
productA.Values = sheet.Range["B2:B6"];
productA.CategoryLabels = sheet.Range["A2:A6"];
//Set second serie
IChartSerie productB = chart.Series.Add("ProductB");
productB.Values = sheet.Range["C2:C6"];
productB.CategoryLabels = sheet.Range["A2:A6"];
//Saving the workbook as stream
FileStream stream = new FileStream("Chart.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Inserts the sample data for the chart
    sheet.Range["A1"].Text = "Month";
    sheet.Range["B1"].Text = "Product A";
    sheet.Range["C1"].Text = "Product B";
    //Months
    sheet.Range["A2"].Text = "Jan";
    sheet.Range["A3"].Text = "Feb";
    sheet.Range["A4"].Text = "Mar";
    sheet.Range["A5"].Text = "Apr";
    sheet.Range["A6"].Text = "May";
    //Create a random Data
    Random r = new Random();
    for (int i = 2; i <= 6; i++)
    {
        for (int j = 2; j <= 3; j++)
        {
            sheet.Range[i, j].Number = r.Next(0, 500);
        }
    }
    IChartShape chart = sheet.Charts.Add();
    //Set chart type
    chart.ChartType = ExcelChartType.Line;
    //Set Chart Title
    chart.ChartTitle = "Product Sales comparison";
}

```

```

//Set first serie
IChartSerie productA = chart.Series.Add("ProductA");
productA.Values = sheet.Range["B2:B6"];
productA.CategoryLabels = sheet.Range["A2:A6"];
//Set second serie
IChartSerie productB = chart.Series.Add("ProductB");
productB.Values = sheet.Range["C2:C6"];
productB.CategoryLabels = sheet.Range["A2:A6"];
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Chart.
xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Chart.xlsx",
"application/msexcel", stream);
}
}

```

Creating a chart Sheet

The following code snippet shows how to create a chart sheet (separate sheet).

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
//Add the chart sheet
IChart chart = workbook.Charts.Add();
chart.ChartType = ExcelChartType.Column_Clustered;
chart.DataRange = sheet.Range["A1:E5"];
workbook.SaveAs("Chart.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)

```

```
'Add the chart sheet
Dim chart As IChart = workbook.Charts.Add()
chart.ChartType = ExcelChartType.Column_Clustered
chart.DataRange = sheet.Range("A1:E5")
workbook.SaveAs("Chart.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Add the chart sheet
    IChart chart = workbook.Charts.Add();
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.DataRange = sheet.Range["A1:E5"];
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Chart";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Add the chart sheet
    IChart chart = workbook.Charts.Add();
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.DataRange = sheet.Range["A1:E5"];
    //Saving the workbook as stream
```

```

FileStream stream = new FileStream("Chart.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    ///Add the chart sheet
    IChart chart = workbook.Charts.Add();
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.DataRange = sheet.Range["A1:E5"];
    ///Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    ///Save the document as file and view the saved document
    ///The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Chart.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Chart.xlsx", "application/msexcel", stream);
    }
}

```

Creating Custom Charts

A custom chart can be created by using different types of charts for different data series.

For example, you can use a column chart for the first data series and a line chart for the second series. As a result you will have a column chart, combined with a line chart.

This sample also explains different chart properties like

[Set Data Range to Chart](#)

C#

```
//Add a new chart with data range
IChartShape chart = sheet.Charts.Add();
chart.DataRange = sheet.Range["A3:C6"];
```

VB.NET

```
'Add a new chart with data range
Dim chart As IChartShape = sheet.Charts.Add()
chart.DataRange = sheet.Range("A3:C6")
```

UWP

```
//Add a new chart with data range
IChartShape chart = sheet.Charts.Add();
chart.DataRange = sheet.Range["A3:C6"];
```

ASP.NET CORE

```
//Add a new chart with data range
IChartShape chart = sheet.Charts.Add();
chart.DataRange = sheet.Range["A3:C6"];
```

XAMARIN

```
//Add a new chart with data range
IChartShape chart = sheet.Charts.Add();
chart.DataRange = sheet.Range["A3:C6"];
```

Name the Chart and Set Chart Title

C#

```
//Set chart name and chart title
chart.Name = "CrescentCity,CA";
chart.ChartTitle = "Crescent City, CA";
```

VB.NET

```
'Set chart name and chart title
chart.Name = "CrescentCity,CA"
chart.ChartTitle = "Crescent City, CA"
```

UWP

```
//Set chart name and chart title
chart.Name = "CrescentCity,CA";
chart.ChartTitle = "Crescent City, CA";
```

ASP.NET CORE

```
//Set chart name and chart title
chart.Name = "CrescentCity,CA";
chart.ChartTitle = "Crescent City, CA";
```


XAMARIN

```
//Set chart name and chart title
chart.Name = "CrescentCity,CA";
chart.ChartTitle = "Crescent City, CA";
```

*Different Primary Value Axis Properties***C#**

```
//Axis title
chart.PrimaryValueAxis.Title = "Precipitation,in.";
//Axis title area text angle rotation
chart.PrimaryValueAxis.TitleArea.TextRotationAngle = 90;
//Maximum value in the axis
chart.PrimaryValueAxis.MaximumValue = 14.0;
//Number format for axis
chart.PrimaryValueAxis.NumberFormat = "0.0";
```

VB.NET

```
'Axis title
chart.PrimaryValueAxis.Title = "Precipitation,in."
'Axis title area text angle rotation
chart.PrimaryValueAxis.TitleArea.TextRotationAngle = 90
'Maximum value in the axis
chart.PrimaryValueAxis.MaximumValue = 14.0
'Number format for axis
chart.PrimaryValueAxis.NumberFormat = "0.0"
```

UWP

```
//Axis title
chart.PrimaryValueAxis.Title = "Precipitation,in.";
//Axis title area text angle rotation
chart.PrimaryValueAxis.TitleArea.TextRotationAngle = 90;
//Maximum value in the axis
chart.PrimaryValueAxis.MaximumValue = 14.0;
//Number format for axis
chart.PrimaryValueAxis.NumberFormat = "0.0";
```

ASP.NET CORE

```
//Axis title
chart.PrimaryValueAxis.Title = "Precipitation,in.";
//Axis title area text angle rotation
chart.PrimaryValueAxis.TitleArea.TextRotationAngle = 90;
//Maximum value in the axis
chart.PrimaryValueAxis.MaximumValue = 14.0;
//Number format for axis
chart.PrimaryValueAxis.NumberFormat = "0.0";
```

XAMARIN

```
//Axis title
chart.PrimaryValueAxis.Title = "Precipitation,in.";
//Axis title area text angle rotation
chart.PrimaryValueAxis.TitleArea.TextRotationAngle = 90;
//Maximum value in the axis
chart.PrimaryValueAxis.MaximumValue = 14.0;
//Number format for axis
chart.PrimaryValueAxis.NumberFormat = "0.0";
```

Different Secondary Value Axis Properties

C#

```
//MaxCross in axis
chart.SecondaryValueAxis.IsMaxCross = true;
//Axis title
chart.SecondaryValueAxis.Title = "Temperature,deg.F";
//Axis title area text angle rotation
chart.SecondaryValueAxis.TitleArea.TextRotationAngle = 90;
```

VB.NET

```
'MaxCross in axis
chart.SecondaryValueAxis.IsMaxCross = true
'Axis title
chart.SecondaryValueAxis.Title = "Temperature,deg.F"
'Axis title area text angle rotation
chart.SecondaryValueAxis.TitleArea.TextRotationAngle = 90
```

UWP

```
//MaxCross in axis
chart.SecondaryValueAxis.IsMaxCross = true;
//Axis title
chart.SecondaryValueAxis.Title = "Temperature,deg.F";
//Axis title area text angle rotation
chart.SecondaryValueAxis.TitleArea.TextRotationAngle = 90;
```

ASP.NET CORE

```
//MaxCross in axis
chart.SecondaryValueAxis.IsMaxCross = true;
//Axis title
chart.SecondaryValueAxis.Title = "Temperature,deg.F";
//Axis title area text angle rotation
chart.SecondaryValueAxis.TitleArea.TextRotationAngle = 90;
```

XAMARIN

```
//MaxCross in axis
chart.SecondaryValueAxis.IsMaxCross = true;
//Axis title
chart.SecondaryValueAxis.Title = "Temperature,deg.F";
//Axis title area text angle rotation
chart.SecondaryValueAxis.TitleArea.TextRotationAngle = 90;
```

*Different Secondary Category Axis Properties***C#**

```
//MaxCross in axis
chart.SecondaryCategoryAxis.IsMaxCross = true;
//Select border line color
chart.SecondaryCategoryAxis.Border.LineColor = Color.Transparent;
//Select major tick mark option
chart.SecondaryCategoryAxis.MajorTickMark = ExcelTickMark.TickMark_None;
//Select tick label position
chart.SecondaryCategoryAxis.TickLabelPosition =
ExcelTickLabelPosition.TickLabelPosition_None;
```

VB.NET

```
'MaxCross in axis
chart.SecondaryCategoryAxis.IsMaxCross = true
'Select border line color
chart.SecondaryCategoryAxis.Border.LineColor = Color.Transparent
'Select major tick mark option
chart.SecondaryCategoryAxis.MajorTickMark = ExcelTickMark.TickMark_None
'Select tick label position
chart.SecondaryCategoryAxis.TickLabelPosition =
ExcelTickLabelPosition.TickLabelPosition_None
```

UWP

```
//MaxCross in axis
chart.SecondaryCategoryAxis.IsMaxCross = true;
//Select border line color
chart.SecondaryCategoryAxis.Border.LineColor = Color.Transparent;
//Select major tick mark option
chart.SecondaryCategoryAxis.MajorTickMark = ExcelTickMark.TickMark_None;
//Select tick label position
chart.SecondaryCategoryAxis.TickLabelPosition =
ExcelTickLabelPosition.TickLabelPosition_None;
```

ASP.NET CORE

```
//MaxCross in axis
chart.SecondaryCategoryAxis.IsMaxCross = true;
//Select border line color
chart.SecondaryCategoryAxis.Border.LineColor = Color.Transparent;
//Select major tick mark option
chart.SecondaryCategoryAxis.MajorTickMark = ExcelTickMark.TickMark_None;
//Select tick label position
chart.SecondaryCategoryAxis.TickLabelPosition =
ExcelTickLabelPosition.TickLabelPosition_None;
```

XAMARIN

```
//MaxCross in axis
chart.SecondaryCategoryAxis.IsMaxCross = true;
```

```
//Select border line color
chart.SecondaryCategoryAxis.Border.LineColor = Color.Transparent;
//Select major tick mark option
chart.SecondaryCategoryAxis.MajorTickMark = ExcelTickMark.TickMark_None;
//Select tick label position
chart.SecondaryCategoryAxis.TickLabelPosition =
ExcelTickLabelPosition.TickLabelPosition_None;
```

Different Chart Series Fill Properties

C#

```
IChartSerie serieOne = chart.Series[0];
//Series name
serieOne.Name = "Precipitation,in.";
//Series fill type
serieOne.SerieFormat.Fill.FillType = ExcelFillType.Gradient;
//Series two color gradient
serieOne.SerieFormat.Fill.TwoColorGradient(ExcelGradientStyle.Vertical,
ExcelGradientVariants.ShadingVariants_2);
//Series gradient color type
serieOne.SerieFormat.Fill.GradientColorType = ExcelGradientColor.TwoColor;
//Series fore color
serieOne.SerieFormat.Fill.ForeColor = Color.Plum;
```

VB.NET

```
Dim serieOne As IChartSerie = chart.Series(0)
'Series name
serieOne.Name = "Precipitation,in."
'Series fill type
serieOne.SerieFormat.Fill.FillType = ExcelFillType.Gradient
'Series two color gradient
serieOne.SerieFormat.Fill.TwoColorGradient(ExcelGradientStyle.Vertical,
ExcelGradientVariants.ShadingVariants_2)
'Series gradient color type
serieOne.SerieFormat.Fill.GradientColorType = ExcelGradientColor.TwoColor
'Series fore color
serieOne.SerieFormat.Fill.ForeColor = Color.Plum
```

UWP

```
IChartSerie serieOne = chart.Series[0];
//Series name
serieOne.Name = "Precipitation,in.";
//Series fill type
serieOne.SerieFormat.Fill.FillType = ExcelFillType.Gradient;
//Series two color gradient
serieOne.SerieFormat.Fill.TwoColorGradient(ExcelGradientStyle.Vertical,
ExcelGradientVariants.ShadingVariants_2);
//Series gradient color type
serieOne.SerieFormat.Fill.GradientColorType = ExcelGradientColor.TwoColor;
//Series fore color
serieOne.SerieFormat.Fill.ForeColor = Color.Plum;
```

ASP.NET CORE

```

IChartSerie serieOne = chart.Series[0];
//Series name
serieOne.Name = "Precipitation,in.";
//Series fill type
serieOne.SerieFormat.Fill.FillType = ExcelFillType.Gradient;
//Series two color gradient
serieOne.SerieFormat.Fill.TwoColorGradient(ExcelGradientStyle.Vertical,
ExcelGradientVariants.ShadingVariants_2);
//Series gradient color type
serieOne.SerieFormat.Fill.GradientColorType = ExcelGradientColor.TwoColor;
//Series fore color
serieOne.SerieFormat.Fill.ForeColor = Color.Plum;

```

XAMARIN

```

IChartSerie serieOne = chart.Series[0];
//Series name
serieOne.Name = "Precipitation,in.";
//Series fill type
serieOne.SerieFormat.Fill.FillType = ExcelFillType.Gradient;
//Series two color gradient
serieOne.SerieFormat.Fill.TwoColorGradient(ExcelGradientStyle.Vertical,
ExcelGradientVariants.ShadingVariants_2);
//Series gradient color type
serieOne.SerieFormat.Fill.GradientColorType = ExcelGradientColor.TwoColor;
//Series fore color
serieOne.SerieFormat.Fill.ForeColor = Color.Plum;

```

*Different Marker Properties***C#**

```

IChartSerie serieTwo = chart.Series[1];
//Marker style
serieTwo.SerieFormat.MarkerStyle = ExcelChartMarkerType.Diamond;
//Marker size
serieTwo.SerieFormat.MarkerSize = 8;
//Marker background color
serieTwo.SerieFormat.MarkerBackgroundColor = Color.DarkGreen;
//Marker foreground color
serieTwo.SerieFormat.MarkerForegroundColor = Color.DarkGreen;

```

VB.NET

```

Dim serieTwo As IChartSerie = chart.Series(1)
'Marker style
serieTwo.SerieFormat.MarkerStyle = ExcelChartMarkerType.Diamond
'Marker size
serieTwo.SerieFormat.MarkerSize = 8
'Marker background color
serieTwo.SerieFormat.MarkerBackgroundColor = Color.DarkGreen
'Marker foreground color
serieTwo.SerieFormat.MarkerForegroundColor = Color.DarkGreen

```

UWP

```
IChartSerie serieTwo = chart.Series[1];  
//Marker style  
serieTwo.SerieFormat.MarkerStyle = ExcelChartMarkerType.Diamond;  
//Marker size  
serieTwo.SerieFormat.MarkerSize = 8;  
//Marker background color  
serieTwo.SerieFormat.MarkerBackgroundColor = Color.DarkGreen;  
//Marker foreground color  
serieTwo.SerieFormat.MarkerForegroundColor = Color.DarkGreen;
```

ASP.NET CORE

```
IChartSerie serieTwo = chart.Series[1];  
//Marker style  
serieTwo.SerieFormat.MarkerStyle = ExcelChartMarkerType.Diamond;  
//Marker size  
serieTwo.SerieFormat.MarkerSize = 8;  
//Marker background color  
serieTwo.SerieFormat.MarkerBackgroundColor = Color.DarkGreen;  
//Marker foreground color  
serieTwo.SerieFormat.MarkerForegroundColor = Color.DarkGreen;
```

XAMARIN

```
IChartSerie serieTwo = chart.Series[1];  
//Marker style  
serieTwo.SerieFormat.MarkerStyle = ExcelChartMarkerType.Diamond;  
//Marker size  
serieTwo.SerieFormat.MarkerSize = 8;  
//Marker background color  
serieTwo.SerieFormat.MarkerBackgroundColor = Color.DarkGreen;  
//Marker foreground color  
serieTwo.SerieFormat.MarkerForegroundColor = Color.DarkGreen;
```

*Different Legend Properties***C#**

```
//Legend without overlapping the chart  
chart.Legend.IncludeInLayout = true;  
//Legend position  
chart.Legend.Position = ExcelLegendPosition.Bottom;  
//View legend horizontally  
chart.Legend.IsVerticalLegend = false;
```

VB.NET

```
'Legend without overlapping the chart  
chart.Legend.IncludeInLayout = true  
'Legend position  
chart.Legend.Position = ExcelLegendPosition.Bottom  
'View legend horizontally  
chart.Legend.IsVerticalLegend = false
```

UWP

```
//Legend without overlapping the chart
chart.Legend.IncludeInLayout = true;
//Legend position
chart.Legend.Position = ExcelLegendPosition.Bottom;
//View legend horizontally
chart.Legend.IsVerticalLegend = false;
```

ASP.NET CORE

```
//Legend without overlapping the chart
chart.Legend.IncludeInLayout = true;
//Legend position
chart.Legend.Position = ExcelLegendPosition.Bottom;
//View legend horizontally
chart.Legend.IsVerticalLegend = false;
```

XAMARIN

```
//Legend without overlapping the chart
chart.Legend.IncludeInLayout = true;
//Legend position
chart.Legend.Position = ExcelLegendPosition.Bottom;
//View legend horizontally
chart.Legend.IsVerticalLegend = false;
```

The complete code snippet illustrating the above options along with creating custom charts is shown below.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Merge cells
    sheet.Range["A1:D1"].Merge();
    //Set Font style as bold
    sheet.Range["A1"].CellStyle.Font.Bold = true;
    //Insert data for the chart
    sheet.Range["A1"].Text = "Crescent City, CA";
    sheet.Range["B3"].Text = "Precipitation,in.";
    sheet.Range["C3"].Text = "Temperature,deg.F";
    sheet.Range["A4"].Text = "Jan";
    sheet.Range["A5"].Text = "Feb";
    sheet.Range["A6"].Text = "March";
    sheet.Range["B4"].Number = 10.9;
    sheet.Range["B5"].Number = 8.9;
    sheet.Range["B6"].Number = 8.6;
    sheet.Range["C4"].Number = 47.5;
    sheet.Range["C5"].Number = 48.7;
```

```

sheet.Range["C6"].Number = 48.9;
//Adjust column width in used range
sheet.UsedRange.AutofitColumns();
//Add a new chart with data range
IChartShape chart = sheet.Charts.Add();
chart.DataRange = sheet.Range["A3:C6"];
//Set chart name and chart title
chart.Name = "CrescentCity,CA";
chart.ChartTitle = "Crescent City, CA";
chart.IsSeriesInRows = false;
//Set primary value axis properties
chart.PrimaryValueAxis.Title = "Precipitation,in.";
chart.PrimaryValueAxis.TitleArea.TextRotationAngle = 90;
chart.PrimaryValueAxis.MaximumValue = 14.0;
chart.PrimaryValueAxis.NumberFormat = "0.0";
//Format first serie fill properties
IChartSerie serieOne = chart.Series[0];
serieOne.Name = "Precipitation,in.";
serieOne.SerieFormat.Fill.FillType = ExcelFillType.Gradient;
serieOne.SerieFormat.Fill.TwoColorGradient(ExcelGradientStyle.Vertical,
ExcelGradientVariants.ShadingVariants_2);
serieOne.SerieFormat.Fill.GradientColorType = ExcelGradientColor.TwoColor;
serieOne.SerieFormat.Fill.ForeColor = Color.Plum;
//Format second serie properties
IChartSerie serieTwo = chart.Series[1];
serieTwo.SerieType = ExcelChartType.Line_Markers;
serieTwo.Name = "Temperature,deg.F";
//Format marker properties
serieTwo.SerieFormat.MarkerStyle = ExcelChartMarkerType.Diamond;
serieTwo.SerieFormat.MarkerSize = 8;
serieTwo.SerieFormat.MarkerBackgroundColor = Color.DarkGreen;
serieTwo.SerieFormat.MarkerForegroundColor = Color.DarkGreen;
serieTwo.SerieFormat.LineProperties.LineColor = Color.DarkGreen;
//Use Secondary Axis
serieTwo.UsePrimaryAxis = false;
//MaxCross for secondary axes
chart.SecondaryCategoryAxis.IsMaxCross = true;
chart.SecondaryValueAxis.IsMaxCross = true;
//Set title for secondary value axis
chart.SecondaryValueAxis.Title = "Temperature,deg.F";
chart.SecondaryValueAxis.TitleArea.TextRotationAngle = 90;
//Set secondary category axis properties
chart.SecondaryCategoryAxis.Border.LineColor = Color.Transparent;
chart.SecondaryCategoryAxis.MajorTickMark = ExcelTickMark.TickMark_None;
chart.SecondaryCategoryAxis.TickLabelPosition =
ExcelTickLabelPosition.TickLabelPosition_None;
//Set legend properties
chart.Legend.Position = ExcelLegendPosition.Bottom;
chart.Legend.IsVerticalLegend = false;
workbook.SaveAs("Chart.xlsx");
}

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Using excelEngine As ExcelEngine = New ExcelEngine()

```



```

Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Merge cells
sheet.Range("A1:D1").Merge()
'Set Font style as bold
sheet.Range("A1").CellStyle.Font.Bold = True
'Insert data for the chart
sheet.Range("A1").Text = "Crescent City, CA"
sheet.Range("B3").Text = "Precipitation,in."
sheet.Range("C3").Text = "Temperature,deg.F"
sheet.Range("A4").Text = "Jan"
sheet.Range("A5").Text = "Feb"
sheet.Range("A6").Text = "March"
sheet.Range("B4").Number = 10.9
sheet.Range("B5").Number = 8.9
sheet.Range("B6").Number = 8.6
sheet.Range("C4").Number = 47.5
sheet.Range("C5").Number = 48.7
sheet.Range("C6").Number = 48.9
'Adjust column width in used range
sheet.UsedRange.AutofitColumns()
'Add a new chart with data range
Dim chart As IChartShape = sheet.Charts.Add()
chart.DataRange = sheet.Range("A3:C6")
'Set chart name and chart title
chart.Name = "CrescentCity,CA"
chart.ChartTitle = "Crescent City, CA"
chart.IsSeriesInRows = False
'Set primary value axis properties
chart.PrimaryValueAxis.Title = "Precipitation,in."
chart.PrimaryValueAxis.TitleArea.TextRotationAngle = 90
chart.PrimaryValueAxis.MaximumValue = 14.0
chart.PrimaryValueAxis.NumberFormat = "0.0"
'Format first serie fill properties
Dim serieOne As IChartSerie = chart.Series(0)
serieOne.Name = "Precipitation,in."
serieOne.SerieFormat.Fill.FillType = ExcelFillType.Gradient
serieOne.SerieFormat.Fill.TwoColorGradient(ExcelGradientStyle.Vertical,
ExcelGradientVariants.ShadingVariants_2)
serieOne.SerieFormat.Fill.GradientColorType = ExcelGradientColor.TwoColor
serieOne.SerieFormat.Fill.ForeColor = Color.Plum
'Format second serie properties
Dim serieTwo As IChartSerie = chart.Series(1)
serieTwo.SerieType = ExcelChartType.Line_Markers
serieTwo.Name = "Temperature,deg.F"
'Format marker properties
serieTwo.SerieFormat.MarkerStyle = ExcelChartMarkerType.Diamond
serieTwo.SerieFormat.MarkerSize = 8
serieTwo.SerieFormat.MarkerBackgroundColor = Color.DarkGreen
serieTwo.SerieFormat.MarkerForegroundColor = Color.DarkGreen
serieTwo.SerieFormat.LineProperties.LineColor = Color.DarkGreen
'Use Secondary Axis
serieTwo.UsePrimaryAxis = False
'MaxCross for secondary axes
chart.SecondaryCategoryAxis.IsMaxCross = True

```

```

chart.SecondaryValueAxis.IsMaxCross = True
'Set title for secondary value axis
chart.SecondaryValueAxis.Title = "Temperature,deg.F"
chart.SecondaryValueAxis.TitleArea.TextRotationAngle = 90
'Set secondary category axis properties
chart.SecondaryCategoryAxis.Border.LineColor = Color.Transparent
chart.SecondaryCategoryAxis.MajorTickMark = ExcelTickMark.TickMark_None
chart.SecondaryCategoryAxis.TickLabelPosition =
ExcelTickLabelPosition.TickLabelPosition_None
'Set legend properties
chart.Legend.Position = ExcelLegendPosition.Bottom
chart.Legend.IsVerticalLegend = False
workbook.SaveAs("Chart.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Merge cells
    sheet.Range["A1:D1"].Merge();
    //Set Font style as bold
    sheet.Range["A1"].CellStyle.Font.Bold = true;
    //Insert data for the chart
    sheet.Range["A1"].Text = "Crescent City, CA";
    sheet.Range["B3"].Text = "Precipitation,in.";
    sheet.Range["C3"].Text = "Temperature,deg.F";
    sheet.Range["A4"].Text = "Jan";
    sheet.Range["A5"].Text = "Feb";
    sheet.Range["A6"].Text = "March";
    sheet.Range["B4"].Number = 10.9;
    sheet.Range["B5"].Number = 8.9;
    sheet.Range["B6"].Number = 8.6;
    sheet.Range["C4"].Number = 47.5;
    sheet.Range["C5"].Number = 48.7;
    sheet.Range["C6"].Number = 48.9;
    //Adjust column width in used range
    sheet.UsedRange.AutofitColumns();
    //Add a new chart with data range
    IChartShape chart = sheet.Charts.Add();
    chart.DataRange = sheet.Range["A3:C6"];
    //Set chart name and chart title
    chart.Name = "CrescentCity,CA";
    chart.ChartTitle = "Crescent City, CA";
    chart.IsSeriesInRows = false;
    //Set primary value axis properties
    chart.PrimaryValueAxis.Title = "Precipitation,in.";
    chart.PrimaryValueAxis.TitleArea.TextRotationAngle = 90;
    chart.PrimaryValueAxis.MaximumValue = 14.0;
    chart.PrimaryValueAxis.NumberFormat = "0.0";
    //Format first serie fill properties
    IChartSerie serieOne = chart.Series[0];
}

```

```

serieOne.Name = "Precipitation,in.";
serieOne.SerieFormat.Fill.FillType = ExcelFillType.Gradient;
serieOne.SerieFormat.Fill.TwoColorGradient(ExcelGradientStyle.Vertical,
ExcelGradientVariants.ShadingVariants_2);
serieOne.SerieFormat.Fill.GradientColorType = ExcelGradientColor.TwoColor;
serieOne.SerieFormat.Fill.ForeColor = Color.FromArgb(255, 221, 160, 221);
//Format second serie properties
IChartSerie serieTwo = chart.Series[1];
serieTwo.SerieType = ExcelChartType.Line_Markers;
serieTwo.Name = "Temperature,deg.F";
//Format marker properties
serieTwo.SerieFormat.MarkerStyle = ExcelChartMarkerType.Diamond;
serieTwo.SerieFormat.MarkerSize = 8;
serieTwo.SerieFormat.MarkerBackgroundColor = Color.FromArgb(255, 0, 100, 0);
serieTwo.SerieFormat.MarkerForegroundColor = Color.FromArgb(255, 0, 100, 0);
serieTwo.SerieFormat.LineProperties.LineColor = Color.FromArgb(255, 0, 100,
0);
//Use Secondary Axis
serieTwo.UsePrimaryAxis = false;
//MaxCross for secondary axes
chart.SecondaryCategoryAxis.IsMaxCross = true;
chart.SecondaryValueAxis.IsMaxCross = true;
//Set title for secondary value axis
chart.SecondaryValueAxis.Title = "Temperature,deg.F";
chart.SecondaryValueAxis.TitleArea.TextRotationAngle = 90;
//Set secondary category axis properties
chart.SecondaryCategoryAxis.Border.LineColor = Color.FromArgb(0, 255, 255,
255);
chart.SecondaryCategoryAxis.MajorTickMark = ExcelTickMark.TickMark_None;
chart.SecondaryCategoryAxis.TickLabelPosition =
ExcelTickLabelPosition.TickLabelPosition_None;
//Set legend properties
chart.Legend.Position = ExcelLegendPosition.Bottom;
chart.Legend.IsVerticalLegend = false;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Chart";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Merge cells
    sheet.Range["A1:D1"].Merge();
}

```

```

//Set Font style as bold
sheet.Range["A1"].CellStyle.Font.Bold = true;
//Insert data for the chart
sheet.Range["A1"].Text = "Crescent City, CA";
sheet.Range["B3"].Text = "Precipitation,in.";
sheet.Range["C3"].Text = "Temperature,deg.F";
sheet.Range["A4"].Text = "Jan";
sheet.Range["A5"].Text = "Feb";
sheet.Range["A6"].Text = "March";
sheet.Range["B4"].Number = 10.9;
sheet.Range["B5"].Number = 8.9;
sheet.Range["B6"].Number = 8.6;
sheet.Range["C4"].Number = 47.5;
sheet.Range["C5"].Number = 48.7;
sheet.Range["C6"].Number = 48.9;
//Adjust column width in used range
sheet.UsedRange.AutofitColumns();
//Add a new chart with data range
IChartShape chart = sheet.Charts.Add();
chart.DataRange = sheet.Range["A3:C6"];
//Set chart name and chart title
chart.Name = "CrescentCity,CA";
chart.ChartTitle = "Crescent City, CA";
chart.IsSeriesInRows = false;
//Set primary value axis properties
chart.PrimaryValueAxis.Title = "Precipitation,in.";
chart.PrimaryValueAxis.TitleArea.TextRotationAngle = 90;
chart.PrimaryValueAxis.MaximumValue = 14.0;
chart.PrimaryValueAxis.NumberFormat = "0.0";
//Format first serie fill properties
IChartSerie serieOne = chart.Series[0];
serieOne.Name = "Precipitation,in.";
serieOne.SerieFormat.Fill.FillType = ExcelFillType.Gradient;
serieOne.SerieFormat.Fill.TwoColorGradient(ExcelGradientStyle.Vertical,
ExcelGradientVariants.ShadingVariants_2);
serieOne.SerieFormat.Fill.GradientColorType = ExcelGradientColor.TwoColor;
serieOne.SerieFormat.Fill.ForeColor = Color.Plum;
//Format second serie properties
IChartSerie serieTwo = chart.Series[1];
serieTwo.SerieType = ExcelChartType.Line_Markers;
serieTwo.Name = "Temperature,deg.F";
//Format marker properties
serieTwo.SerieFormat.MarkerStyle = ExcelChartMarkerType.Diamond;
serieTwo.SerieFormat.MarkerSize = 8;
serieTwo.SerieFormat.MarkerBackgroundColor = Color.DarkGreen;
serieTwo.SerieFormat.MarkerForegroundColor = Color.DarkGreen;
serieTwo.SerieFormat.LineProperties.LineColor = Color.DarkGreen;
//Use Secondary Axis
serieTwo.UsePrimaryAxis = false;
//MaxCross for secondary axes
chart.SecondaryCategoryAxis.IsMaxCross = true;
chart.SecondaryValueAxis.IsMaxCross = true;
//Set title for secondary value axis
chart.SecondaryValueAxis.Title = "Temperature,deg.F";
chart.SecondaryValueAxis.TitleArea.TextRotationAngle = 90;
//Set secondary category axis properties
chart.SecondaryCategoryAxis.Border.LineColor = Color.Transparent;

```

```

chart.SecondaryCategoryAxis.MajorTickMark = ExcelTickMark.TickMark_None;
chart.SecondaryCategoryAxis.TickLabelPosition =
ExcelTickLabelPosition.TickLabelPosition_None;
//Set legend properties
chart.Legend.Position = ExcelLegendPosition.Bottom;
chart.Legend.IsVerticalLegend = false;
//Saving the workbook as stream
FileStream stream = new FileStream("Chart.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

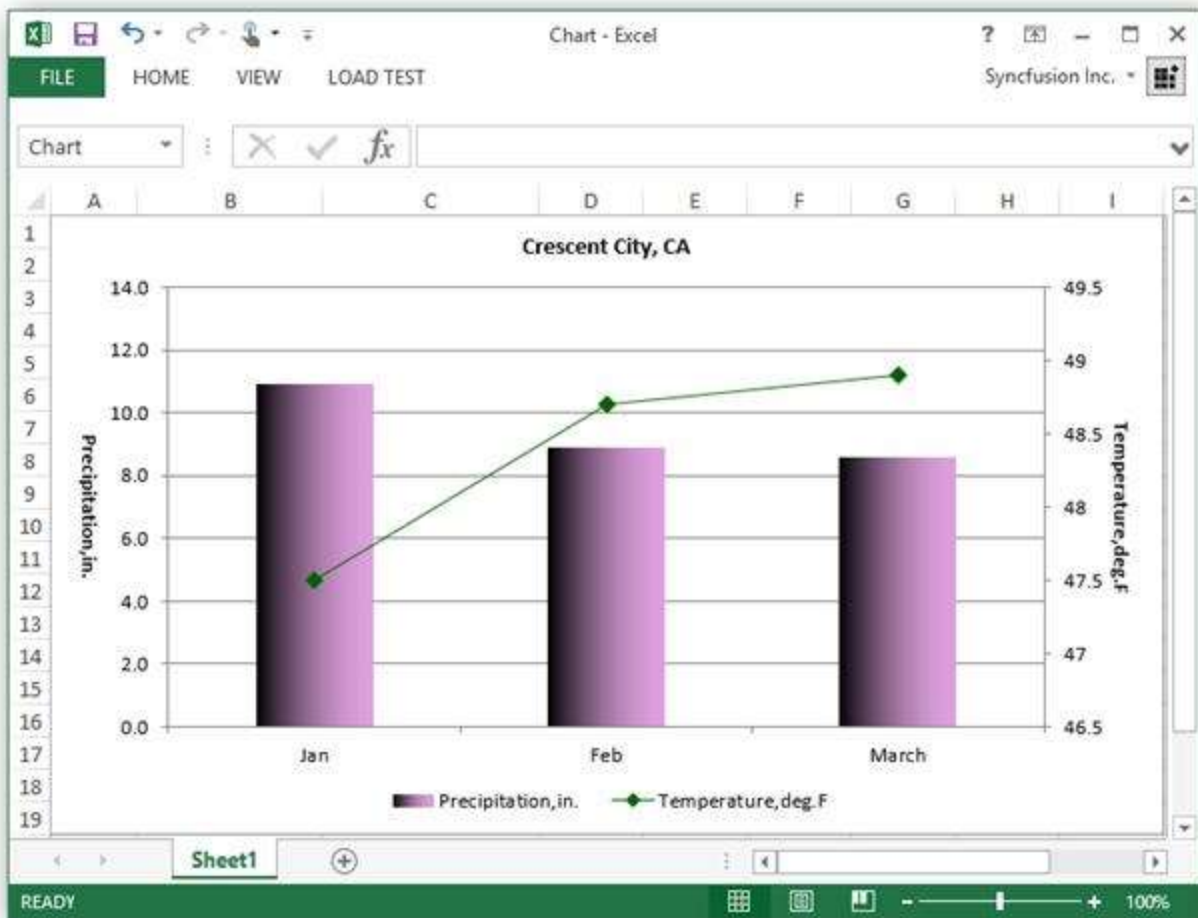
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Merge cells
    sheet.Range["A1:D1"].Merge();
    //Set Font style as bold
    sheet.Range["A1"].CellStyle.Font.Bold = true;
    //Insert data for the chart
    sheet.Range["A1"].Text = "Crescent City, CA";
    sheet.Range["B3"].Text = "Precipitation,in.";
    sheet.Range["C3"].Text = "Temperature,deg.F";
    sheet.Range["A4"].Text = "Jan";
    sheet.Range["A5"].Text = "Feb";
    sheet.Range["A6"].Text = "March";
    sheet.Range["B4"].Number = 10.9;
    sheet.Range["B5"].Number = 8.9;
    sheet.Range["B6"].Number = 8.6;
    sheet.Range["C4"].Number = 47.5;
    sheet.Range["C5"].Number = 48.7;
    sheet.Range["C6"].Number = 48.9;
    //Adjust column width in used range
    sheet.UsedRange.AutofitColumns();
    //Add a new chart with data range
    IChartShape chart = sheet.Charts.Add();
    chart.DataRange = sheet.Range["A3:C6"];
    //Set chart name and chart title
    chart.Name = "CrescentCity,CA";
    chart.ChartTitle = "Crescent City, CA";
    chart.IsSeriesInRows = false;
    //Set primary value axis properties
    chart.PrimaryValueAxis.Title = "Precipitation,in.";
    chart.PrimaryValueAxis.TitleArea.TextRotationAngle = 90;
    chart.PrimaryValueAxis.MaximumValue = 14.0;
    chart.PrimaryValueAxis.NumberFormat = "0.0";
    //Format first serie fill properties
    IChartSerie serieOne = chart.Series[0];
    serieOne.Name = "Precipitation,in.";
    serieOne.SerieFormat.Fill.FillType = ExcelFillType.Gradient;
}

```

```

serieOne.SerieFormat.Fill.TwoColorGradient(ExcelGradientStyle.Vertical,
ExcelGradientVariants.ShadingVariants_2);
serieOne.SerieFormat.Fill.GradientColorType = ExcelGradientColor.TwoColor;
serieOne.SerieFormat.Fill.ForeColor = Syncfusion.Drawing.Color.Plum;
//Format second serie properties
IChartSerie serieTwo = chart.Series[1];
serieTwo.SerieType = ExcelChartType.Line_Markers;
serieTwo.Name = "Temperature,deg.F";
//Format marker properties
serieTwo.SerieFormat.MarkerStyle = ExcelChartMarkerType.Diamond;
serieTwo.SerieFormat.MarkerSize = 8;
serieTwo.SerieFormat.MarkerBackgroundColor =
Syncfusion.Drawing.Color.DarkGreen;
serieTwo.SerieFormat.MarkerForegroundColor =
Syncfusion.Drawing.Color.DarkGreen;
serieTwo.SerieFormat.LineProperties.LineColor =
Syncfusion.Drawing.Color.DarkGreen;
//Use Secondary Axis
serieTwo.UsePrimaryAxis = false;
//MaxCross for secondary axes
chart.SecondaryCategoryAxis.IsMaxCross = true;
chart.SecondaryValueAxis.IsMaxCross = true;
//Set title for secondary value axis
chart.SecondaryValueAxis.Title = "Temperature,deg.F";
chart.SecondaryValueAxis.TitleArea.TextRotationAngle = 90;
//Set secondary category axis properties
chart.SecondaryCategoryAxis.Border.LineColor =
Syncfusion.Drawing.Color.Transparent;
chart.SecondaryCategoryAxis.MajorTickMark = ExcelTickMark.TickMark_None;
chart.SecondaryCategoryAxis.TickLabelPosition =
ExcelTickLabelPosition.TickLabelPosition_None;
//Set legend properties
chart.Legend.Position = ExcelLegendPosition.Bottom;
chart.Legend.IsVerticalLegend = false;
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Chart.
xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Chart.xlsx",
"application/msexcel", stream);
}
}

```



Remove a chart

The following code snippet shows how to remove the chart from the worksheet using **Remove** method.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    IChartShape chart = sheet.Charts[0];
    //Remove the chart from the worksheet
    chart.Remove();
    workbook.SaveAs("Chart.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
```

```

Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim chart As IChartShape = sheet.Charts(0)
'Remove the chart from the worksheet
chart.Remove()
workbook.SaveAs("Chart.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    IChartShape chart = sheet.Charts[0];
    //Remove the chart from the worksheet
    chart.Remove();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Chart";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    IChartShape chart = sheet.Charts[0];
    //Remove the chart from the worksheet
    chart.Remove();
    //Saving the workbook as stream

```



```

FileStream stream = new FileStream("Chart.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    IChartShape chart = sheet.Charts[0];
    ///Remove the chart from the worksheet
    chart.Remove();
    ///Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    ///Save the document as file and view the saved document
    ///The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Chart.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Chart.xlsx", "application/msexcel", stream);
    }
}

```

Chart Appearance Settings

The appearance of a chart can be modified according to the convenience and usage.

Elements of Chart

The following screen shot shows the elements of chart.



1. The chart area of the chart.
2. The plot area of the chart.
3. The data points of the data series that are plotted in the chart.
4. The horizontal (category) and vertical (value) axis along which the data is plotted in the chart.
5. The legend of the chart.
6. A chart axis title that you can use in the chart.
7. A data label that you can use to identify the details of a data point in a data series.

Chart Area Appearance

The following code snippet shows how to modify the appearance of the chart area.

C#

```
//Format Chart Area
IChartFrameFormat chartArea = chart.ChartArea;
//Chart Area Settings
chartArea.Fill.FillType = ExcelFillType.Gradient;
//Set Fill Effects
chartArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartArea.Fill.ForeColor = Color.White;
```

VB.NET

```
'Format Chart Area
Dim chartArea As IChartFrameFormat = chart.ChartArea
'Chart Area Settings
chartArea.Fill.FillType = ExcelFillType.Gradient
'Set Fill Effects
chartArea.Fill.BackColor = Color.FromArgb(205, 217, 234)
chartArea.Fill.ForeColor = Color.White
```

UWP

```
//Format Chart Area
IChartFrameFormat chartArea = chart.ChartArea;
//Chart Area Settings
chartArea.Fill.FillType = ExcelFillType.Gradient;
//Set Fill Effects
chartArea.Fill.BackColor = Color.FromArgb(255, 205, 217, 234);
chartArea.Fill.ForeColor = Color.FromArgb(255, 255, 255, 255);
```

ASP.NET CORE

```
//Format Chart Area
IChartFrameFormat chartArea = chart.ChartArea;
//Chart Area Settings
chartArea.Fill.FillType = ExcelFillType.Gradient;
//Set Fill Effects
chartArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartArea.Fill.ForeColor = Color.White;
```

XAMARIN

```
//Format Chart Area
IChartFrameFormat chartArea = chart.ChartArea;
//Chart Area Settings
chartArea.Fill.FillType = ExcelFillType.Gradient;
//Set Fill Effects
chartArea.Fill.BackColor = Syncfusion.Drawing.Color.FromArgb(205, 217, 234);
chartArea.Fill.ForeColor = Syncfusion.Drawing.Color.White;
```

Plot Area Appearance

The following code snippet shows how to modify the appearance of the plot area.

C#

```
//Set Plot Area
IChartFrameFormat chartPlotArea = chart.PlotArea;
//Set fill color
chartPlotArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartPlotArea.Fill.ForeColor = Color.White;
```

VB.NET

```
'Set Plot Area
Dim chartPlotArea As IChartFrameFormat = chart.PlotArea
'Set fill color
chartPlotArea.Fill.BackColor = Color.FromArgb(205, 217, 234)
chartPlotArea.Fill.ForeColor = Color.White
```

UWP

```
//Set Plot Area
IChartFrameFormat chartPlotArea = chart.PlotArea;
//Set fill color
chartPlotArea.Fill.BackColor = Color.FromArgb(255, 205, 217, 234);
chartPlotArea.Fill.ForeColor = Color.FromArgb(255, 255, 255, 255);
```

ASP.NET CORE

```
//Set Plot Area
IChartFrameFormat chartPlotArea = chart.PlotArea;
//Set fill color
chartPlotArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartPlotArea.Fill.ForeColor = Color.White;
```

XAMARIN

```
//Set Plot Area
IChartFrameFormat chartPlotArea = chart.PlotArea;
//Set fill color
chartPlotArea.Fill.BackColor = Syncfusion.Drawing.Color.FromArgb(205, 217, 234);
chartPlotArea.Fill.ForeColor = Syncfusion.Drawing.Color.White;
```

Data Labels Appearance

The following code snippet illustrates how to modify the appearance of data labels.

C#

```
IChartSerie serie = chart.Series[0];
//Set data labels color
serie.DataPoints.DefaultDataPoint.DataLabels.Color = ExcelKnownColors.Blue;
serie.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
```

VB.NET

```
Dim serie As IChartSerie = chart.Series(0)
'Set data labels color
serie.DataPoints.DefaultDataPoint.DataLabels.Color = ExcelKnownColors.Blue
serie.DataPoints.DefaultDataPoint.DataLabels.IsValue = True
```

UWP

```
IChartSerie serie = chart.Series[0];
//Set data labels color
serie.DataPoints.DefaultDataPoint.DataLabels.Color = ExcelKnownColors.Blue;
serie.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
```

ASP.NET CORE

```
IChartSerie serie = chart.Series[0];
//Set data labels color
serie.DataPoints.DefaultDataPoint.DataLabels.Color = ExcelKnownColors.Blue;
serie.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
```

XAMARIN

```
IChartSerie serie = chart.Series[0];
//Set data labels color
```

```
serie.DataPoints.DefaultDataPoint.DataLabels.Color = ExcelKnownColors.Blue;
serie.DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
```

Series Appearance

The following code snippet illustrates how to modify the appearance of chart series.

C#

```
IChartSerie serie = chart.Series[0];
//Fill Effects
serie.SerieFormat.Fill.FillType = ExcelFillType.Gradient;
serie.SerieFormat.Fill.ForeColor = Color.Yellow;
```

VB.NET

```
Dim serie As IChartSerie = chart.Series(0)
'Fill Effects
serie.SerieFormat.Fill.FillType = ExcelFillType.Gradient
serie.SerieFormat.Fill.ForeColor = Color.Yellow
```

UWP

```
IChartSerie serie = chart.Series[0];
//Fill Effects
serie.SerieFormat.Fill.FillType = ExcelFillType.Gradient;
serie.SerieFormat.Fill.ForeColor = Color.FromArgb(255, 255, 255, 0);
```

ASP.NET CORE

```
IChartSerie serie = chart.Series[0];
//Fill Effects
serie.SerieFormat.Fill.FillType = ExcelFillType.Gradient;
serie.SerieFormat.Fill.ForeColor = Color.Yellow;
```

XAMARIN

```
IChartSerie serie = chart.Series[0];
//Fill Effects
serie.SerieFormat.Fill.FillType = ExcelFillType.Gradient;
serie.SerieFormat.Fill.ForeColor = Syncfusion.Drawing.Color.Yellow;
```

The complete code snippet illustrating the above options is shown below.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    IChartShape chart = sheet.Charts.Add();
    chart.DataRange = sheet.UsedRange;
```

```

//Format Chart Area
IChartFrameFormat chartArea = chart.ChartArea;
//Fill Effects
chartArea.Fill.FillType = ExcelFillType.Gradient;
//Set chart area fill color
chartArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartArea.Fill.ForeColor = Color.WhiteSmoke;
//Format Plot Area
IChartFrameFormat chartPlotArea = chart.PlotArea;
//Fill Effects
chartPlotArea.Fill.FillType = ExcelFillType.Gradient;
//Set plot area fill color
chartPlotArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartPlotArea.Fill.ForeColor = Color.YellowGreen;
workbook.SaveAs("Chart.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim chart As IChartShape = sheet.Charts.Add()
chart.DataRange = sheet.UsedRange
'Format Chart Area
Dim chartArea As IChartFrameFormat = chart.ChartArea
'Fill Effects
chartArea.Fill.FillType = ExcelFillType.Gradient
'Set chart area fill color
chartArea.Fill.BackColor = Color.FromArgb(205, 217, 234)
chartArea.Fill.ForeColor = Color.White
'Format Plot Area
Dim chartPlotArea As IChartFrameFormat = chart.PlotArea
'Fill Effects
chartPlotArea.Fill.FillType = ExcelFillType.Gradient
'Set plot area fill color
chartPlotArea.Fill.BackColor = Color.FromArgb(205, 217, 234)
chartPlotArea.Fill.ForeColor = Color.White
workbook.SaveAs("Chart.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
}

```

```

//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
IChartShape chart = sheet.Charts.Add();
chart.DataRange = sheet.UsedRange;
//Format Chart Area
IChartFrameFormat chartArea = chart.ChartArea;
//Fill Effects
chartArea.Fill.FillType = ExcelFillType.Gradient;
//Set chart area fill color
chartArea.Fill.BackColor = Color.FromArgb(255, 205, 217, 234);
chartArea.Fill.ForeColor = Color.FromArgb(255, 245, 245, 245);
//Format Plot Area
IChartFrameFormat chartPlotArea = chart.PlotArea;
//Fill Effects
chartPlotArea.Fill.FillType = ExcelFillType.Gradient;
//Set plot area fill color
chartPlotArea.Fill.BackColor = Color.FromArgb(255, 205, 217, 234);
chartPlotArea.Fill.ForeColor = Color.FromArgb(255, 154, 205, 50);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Chart";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
IChartShape chart = sheet.Charts.Add();
chart.DataRange = sheet.UsedRange;
//Format Chart Area
IChartFrameFormat chartArea = chart.ChartArea;
//Fill Effects
chartArea.Fill.FillType = ExcelFillType.Gradient;
//Set chart area fill color
chartArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartArea.Fill.ForeColor = Color.WhiteSmoke;
//Format Plot Area
IChartFrameFormat chartPlotArea = chart.PlotArea;
//Fill Effects
chartPlotArea.Fill.FillType = ExcelFillType.Gradient;

```

```
//Set plot area fill color
chartPlotArea.Fill.BackColor = Color.FromArgb(205, 217, 234);
chartPlotArea.Fill.ForeColor = Color.YellowGreen;
//Saving the workbook as stream
FileStream stream = new FileStream("Chart.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    IChartShape chart = sheet.Charts.Add();
    chart.DataRange = sheet.UsedRange;
    //Format Chart Area
    IChartFrameFormat chartArea = chart.ChartArea;
    //Fill Effects
    chartArea.Fill.FillType = ExcelFillType.Gradient;
    //Set chart area fill color
    chartArea.Fill.BackColor = Syncfusion.Drawing.Color.FromArgb(205, 217, 234);
    chartArea.Fill.ForeColor = Syncfusion.Drawing.Color.WhiteSmoke;
    //Format Plot Area
    IChartFrameFormat chartPlotArea = chart.PlotArea;
    //Fill Effects
    chartPlotArea.Fill.FillType = ExcelFillType.Gradient;
    //Set plot area fill color
    chartPlotArea.Fill.BackColor = Syncfusion.Drawing.Color.FromArgb(205, 217, 234);
    chartPlotArea.Fill.ForeColor = Syncfusion.Drawing.Color.YellowGreen;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone, Android and iOS platforms. Please refer xlsio/xamarin section for respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Chart.xlsx", "application/msexcel", stream);
    }
    else

```



```
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Chart.xlsx",
    "application/msexcel", stream);
}
```

Font settings for chart legend and data labels

Essential XlsIO allows you to set the desired font to legend and series data labels for legend through [TextArea](#) in [IChartLegend](#). Similarly, desired font for data labels of chart series can be set through [DataLabels](#) in [IChartDataPoints](#).

The font style includes font name, font size and font color which are set through [FontName](#), [Size](#) and [Color](#) properties respectively.

Refer the following complete code snippets.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Adding a chart in Excel worksheet
    IChartShape chart = sheet.Charts.Add();
    chart.DataRange = sheet.Range["A1:B5"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Displaying the data label values of chart series
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    chart.Series[1].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    //Setting font name, size and color for chart legend
    chart.Legend.TextArea.FontName = "Tahoma";
    chart.Legend.TextArea.Size = 20;
    chart.Legend.TextArea.Color = ExcelKnownColors.Red;
    //Setting font name, size and color for data labels of first series
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.FontName = "Tahoma";
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 20;
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Color =
    ExcelKnownColors.Red;
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding a chart in Excel worksheet
Dim chart As IChartShape = worksheet.Charts.Add
chart.DataRange = worksheet.Range("A1:B5")
chart.ChartType = ExcelChartType.Column_Clustered
chart.IsSeriesInRows = False
```

```

'Displaying the data label values of chart series
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.IsValue = True
chart.Series(1).DataPoints.DefaultDataPoint.DataLabels.IsValue = True
'Setting font name, size and color for chart legend
chart.Legend.TextArea.FontName = "Tahoma"
chart.Legend.TextArea.Size = 20
chart.Legend.TextArea.Color = ExcelKnownColors.Red
'Setting font name, size and color for data labels of first series
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.FontName = "Tahoma"
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.Size = 20
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.Color =
ExcelKnownColors.Red
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the file picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opening an existing workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding a chart in Excel worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:B5"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Displaying the data label values of chart series
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    chart.Series[1].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    //Setting font name, size and color for chart legend
    chart.Legend.TextArea.FontName = "Tahoma";
    chart.Legend.TextArea.Size = 20;
    chart.Legend.TextArea.Color = ExcelKnownColors.Red;
    //Setting font name, size and color for data labels of first series
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.FontName = "Tahoma";
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 20;
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Color =
ExcelKnownColors.Red;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
}

```

```
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding a chart in Excel worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:B5"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Displaying the data label values of chart series
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    chart.Series[1].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    //Setting font name, size and color for chart legend
    chart.Legend.TextArea.FontName = "Tahoma";
    chart.Legend.TextArea.Size = 20;
    chart.Legend.TextArea.Color = ExcelKnownColors.Red;
    //Setting font name, size and color for data labels of first series
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.FontName = "Tahoma";
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 20;
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Color =
    ExcelKnownColors.Red;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    // "App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding a chart in Excel worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:B5"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Displaying the data label values of chart series
```

```

chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
chart.Series[1].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
//Setting font name, size and color for chart legend
chart.Legend.TextArea.FontName = "Tahoma";
chart.Legend.TextArea.Size = 20;
chart.Legend.TextArea.Color = ExcelKnownColors.Red;
//Setting font name, size and color for data labels of first series
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.FontName = "Tahoma";
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 20;
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Color =
ExcelKnownColors.Red;
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Border Style for Chart Series

A unique border style like line color, line weight, and line pattern can be set for each chart series. Also, these settings can be made for each data point in the chart series.

Refer the following complete code snippets.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
//Adding chart in the worksheet
IChartShape chart = worksheet.Charts.Add();
chart.DataRange = worksheet.Range["A1:B5"];
chart.ChartType = ExcelChartType.Column_Clustered;
chart.IsSeriesInRows = false;
//Accessing first chart series
IChartSerie serie = chart.Series[0];
//Formatting the series border
serie.SerieFormat.LineProperties.LineColor = Color.Brown;
}

```

```

serie.SerieFormat.LineProperties.LinePattern =
ExcelChartLinePattern.CircleDot;
serie.SerieFormat.LineProperties.LineWeight = ExcelChartLineWeight.Wide;
workbook.SaveAs ("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding chart in the worksheet
Dim chart As IChartShape = worksheet.Charts.Add
chart.DataRange = worksheet.Range("A1:B5")
chart.ChartType = ExcelChartType.Column_Clustered
chart.IsSeriesInRows = False
'Accessing first chart series
Dim serie As IChartSerie = chart.Series(0)
'Formatting the series border
serie.SerieFormat.LineProperties.LineColor = Color.Brown
serie.SerieFormat.LineProperties.LinePattern =
ExcelChartLinePattern.CircleDot
serie.SerieFormat.LineProperties.LineWeight = ExcelChartLineWeight.Wide
workbook.SaveAs ("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the file picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opening an existing workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Adding chart in the worksheet
IChartShape chart = worksheet.Charts.Add();
chart.DataRange = worksheet.Range["A1:B5"];
chart.ChartType = ExcelChartType.Column_Clustered;
chart.IsSeriesInRows = false;
//Accessing first chart series
IChartSerie serie = chart.Series[0];
//Formatting the series border
serie.SerieFormat.LineProperties.LineColor = Color.FromArgb(225, 165, 42,
42);
serie.SerieFormat.LineProperties.LinePattern =
ExcelChartLinePattern.CircleDot;
serie.SerieFormat.LineProperties.LineWeight = ExcelChartLineWeight.Wide;

```

```
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:B5"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Accessing first chart series
    IChartSerie serie = chart.Series[0];
    //Formatting the series border
    serie.SerieFormat.LineProperties.LineColor = Color.Brown;
    serie.SerieFormat.LineProperties.LinePattern =
    ExcelChartLinePattern.CircleDot;
    serie.SerieFormat.LineProperties.LineWeight = ExcelChartLineWeight.Wide;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    // "App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the worksheet
    IChartShape chart = worksheet.Charts.Add();
```

```

chart.DataRange = worksheet.Range["A1:B5"];
chart.ChartType = ExcelChartType.Column_Clustered;
chart.IsSeriesInRows = false;
//Accessing first chart series
IChartSerie serie = chart.Series[0];
//Formatting the series border
serie.SerieFormat.LineProperties.LineColor = Syncfusion.Drawing.Color.Brown;
serie.SerieFormat.LineProperties.LinePattern =
ExcelChartLinePattern.CircleDot;
serie.SerieFormat.LineProperties.LineWeight = ExcelChartLineWeight.Wide;
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Adjust space between chart bars

Spaces between chart bars are of two types.

1. **Series Overlap** : Space between bars of different data series of single category.
2. **Gap Width** : Space between different categories.

Essential XlsIO allows you to adjust the space between chart bars using [Overlap](#) and [GapWidth](#) properties of **IChartFormat** interface.

Refer the following complete code snippets.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
//Adding chart in the worksheet
IChartShape chart = worksheet.Charts.Add();
chart.DataRange = worksheet.Range["A1:B5"];
chart.ChartType = ExcelChartType.Column_Clustered;
chart.IsSeriesInRows = false;
}

```

```
//Adding space between bars of different series of single category
chart.Series[0].SerieFormat.CommonSerieOptions.Overlap = 60;
//Adding space between bars of different categories
chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 80;
workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding chart in the worksheet
Dim chart As IChartShape = worksheet.Charts.Add
chart.DataRange = worksheet.Range("A1:B5")
chart.ChartType = ExcelChartType.Column_Clustered
chart.IsSeriesInRows = False
'Adding space between bars of different series of single category
chart.Series(0).SerieFormat.CommonSerieOptions.Overlap = 60
'Adding space between bars of different categories
chart.Series(0).SerieFormat.CommonSerieOptions.GapWidth = 80
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the file picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opening an existing workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Adding chart in the worksheet
IChartShape chart = worksheet.Charts.Add();
chart.DataRange = worksheet.Range["A1:B5"];
chart.ChartType = ExcelChartType.Column_Clustered;
chart.IsSeriesInRows = false;
//Adding space between bars of different series of single category
chart.Series[0].SerieFormat.CommonSerieOptions.Overlap = 60;
//Adding space between bars of different categories
chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 80;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
}
```



```
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:B5"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Adding space between bars of different series of single category
    chart.Series[0].SerieFormat.CommonSerieOptions.Overlap = 60;
    //Adding space between bars of different categories
    chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 80;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    // "App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:B5"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Adding space between bars of different series of single category
    chart.Series[0].SerieFormat.CommonSerieOptions.Overlap = 60;
    //Adding space between bars of different categories
    chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 80;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
}
```

```

workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android, and iOS platforms. Refer to xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Hide Chart Gridlines

Excel chart consists of two types of gridlines such as **major gridlines** and **minor gridlines**. Major gridlines represent the main values in the axis and minor gridlines represent possible values between two adjacent axis values. You can show or hide these gridlines using [HasMajorGridlines](#) and [HasMinorGridlines](#) of **IXChartAxis** interface.

Essential XlsIO supports formatting of gridlines as well through the [MajorGridlines](#) and [MinorGridlines](#) of **IXChartAxis**.

Refer the following complete code snippets.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the Excel worksheet
    IXChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:B5"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Hiding major gridlines
    chart.PrimaryValueAxis.HasMajorGridLines = false;
    //Showing minor gridlines
    chart.PrimaryValueAxis.HasMinorGridLines = true;
    workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")

```

```

Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding chart in the Excel worksheet
Dim chart As IChartShape = worksheet.Charts.Add
chart.DataRange = worksheet.Range("A1:B5")
chart.ChartType = ExcelChartType.Column_Clustered
chart.IsSeriesInRows = False
'Hiding major gridlines
chart.PrimaryValueAxis.HasMajorGridLines = False
'Showing minor gridlines
chart.PrimaryValueAxis.HasMinorGridLines = True
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the file picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opening an existing workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the Excel worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:B5"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Hiding major gridlines
    chart.PrimaryValueAxis.HasMajorGridLines = false;
    //Showing minor gridlines
    chart.PrimaryValueAxis.HasMinorGridLines = true;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;

```

```

FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream);
IWorksheet worksheet = workbook.Worksheets[0];
//Adding chart in the Excel worksheet
IChartShape chart = worksheet.Charts.Add();
chart.DataRange = worksheet.Range["A1:B5"];
chart.ChartType = ExcelChartType.Column_Clustered;
chart.IsSeriesInRows = false;
//Hiding major gridlines
chart.PrimaryValueAxis.HasMajorGridLines = false;
//Showing minor gridlines
chart.PrimaryValueAxis.HasMinorGridLines = true;
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    /"App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
        ple.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the Excel worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:B5"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Hiding major gridlines
    chart.PrimaryValueAxis.HasMajorGridLines = false;
    //Showing minor gridlines
    chart.PrimaryValueAxis.HasMinorGridLines = true;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Refer to the xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
}

```

```
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
    "application/msexcel", stream);
}
}
```

Add High-Low Lines

High-low lines are used in Excel line charts and stock charts that connect the highest and lowest points of a category.

The following code snippet shows how to add High-low lines in a stock chart.

C#

```
using (ExcelEngine engine = new ExcelEngine())
{
    IApplication application = engine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    IChartShape chart = worksheet.Charts[0];
    IChartSerie chartSerie = chart.Series[0];
    //Set HasHighLowLines property to true.
    chartSerie.SerieFormat.CommonSerieOptions.HasHighLowLines = true;
    //Apply formats to HighLowLines.
    chartSerie.SerieFormat.CommonSerieOptions.HighLowLines.LineColor =
    Color.Blue;
    workbook.SaveAs("HighLowLines.xlsx");
    workbook.Close();
}
```

VB.NET

```
Using engine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = engine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim chart As IChartShape = worksheet.Charts(0)
Dim chartSerie As IChartSerie = chart.Series(0)
'Set HasHighLowLines property to true.
chartSerie.SerieFormat.CommonSerieOptions.HasHighLowLines = True;
'Apply formats to HighLowLines.
chartSerie.SerieFormat.CommonSerieOptions.HighLowLines.LineColor =
Color.Blue
workbook.SaveAs("HighLowLines.xlsx")
workbook.Close()
End Using
```

UWP

```
using (ExcelEngine engine = new ExcelEngine())
{
    IApplication application = engine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the file picker
}
```

```

FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opening an existing workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
IChartShape chart = worksheet.Charts[0];
IChartSerie chartSerie = chart.Series[0];
//Set HasHighLowLines property to true.
chartSerie.SerieFormat.CommonSerieOptions.HasHighLowLines = true;
//Apply formats to HighLowLines.
chartSerie.SerieFormat.CommonSerieOptions.HighLowLines.LineColor =
Color.Blue;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "HighLowLines";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine engine = new ExcelEngine())
{
    IApplication application = engine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    IChartShape chart = worksheet.Charts[0];
    IChartSerie chartSerie = chart.Series[0];
    //Set HasHighLowLines property to true.
    chartSerie.SerieFormat.CommonSerieOptions.HasHighLowLines = true;
    //Apply formats to HighLowLines.
    chartSerie.SerieFormat.CommonSerieOptions.HighLowLines.LineColor = Color.
    Blue;
    FileStream stream = new FileStream("HighLowLines.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
    workbook.Close();
}

```

XAMARIN

```

using (ExcelEngine engine = new ExcelEngine())
{
    IApplication application = engine.Excel;

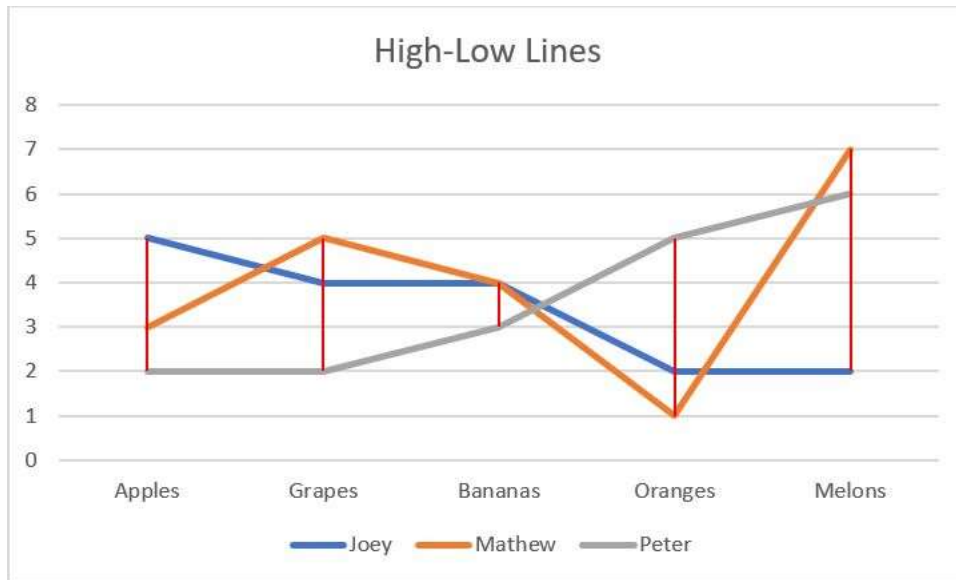
```

```

application.DefaultVersion = ExcelVersion.Excel2013;
// "App" is the class of portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet worksheet = workbook.Worksheets[0];
IChartShape chart = worksheet.Charts[0];
IChartSerie chartSerie = chart.Series[0];
// Set HasHighLowLines property to true.
chartSerie.SerieFormat.CommonSerieOptions.HasHighLowLines = true;
// Apply formats to HighLowLines.
chartSerie.SerieFormat.CommonSerieOptions.HighLowLines.LineColor =
Color.Blue;
// Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
// Save the document as file and view the saved document
// The operation in SaveAndView under Xamarin varies between Windows Phone,
// Android and iOS platforms. Refer to the xlsio/xamarin section for respective
// code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("HighLowLines.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("HighLowLines.xlsx", "application/msexcel", stream);
}
workbook.Close();
}

```

The following screen shot shows the high-low lines in the line chart.



Add Drop Lines

Drop lines are used in Excel area and line charts that helps viewers to determine the data point down to the horizontal axis.

The following code snippet shows how to add Drop lines in a stock chart.

C#

```
using (ExcelEngine engine = new ExcelEngine())
{
    IApplication application = engine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    IChartShape chart = worksheet.Charts[0];
    IChartSerie chartSerie = chart.Series[0];
    //Set HasDropLines property to true.
    chartSerie.SeriesFormat.CommonSeriesOptions.HasDropLines = true;
    //Apply formats to DropLines.
    chartSerie.SeriesFormat.CommonSeriesOptions.DropLines.LineColor = Color.Green;
    workbook.SaveAs("DropLines.xlsx");
    workbook.Close();
}
```

VB.NET

```
Using engine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = engine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim chart As IChartShape = worksheet.Charts(0)
Dim chartSerie As IChartSerie = chart.Series(0)
'Set HasDropLines property to true.
chartSerie.SeriesFormat.CommonSeriesOptions.HasDropLines = True;
'Apply formats to DropLines.
chartSerie.SeriesFormat.CommonSeriesOptions.DropLines.LineColor = Color.Green
workbook.SaveAs("DropLines.xlsx")
workbook.Close()
```


End Using**UWP**

```

using (ExcelEngine engine = new ExcelEngine())
{
    IApplication application = engine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the file picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opening an existing workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    IChartShape chart = worksheet.Charts[0];
    IChartSerie chartSerie = chart.Series[0];
    //Set HasDropLines property to true.
    chartSerie.SerieFormat.CommonSerieOptions.HasDropLines = true;
    //Apply formats to DropLines.
    chartSerie.SerieFormat.CommonSerieOptions.DropLines.LineColor = Color.Green;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "DropLines";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine engine = new ExcelEngine())
{
    IApplication application = engine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    IChartShape chart = worksheet.Charts[0];
    IChartSerie chartSerie = chart.Series[0];
    //Set HasDropLines property to true.
    chartSerie.SerieFormat.CommonSerieOptions.HasDropLines = true;
    //Apply formats to DropLines.
    chartSerie.SerieFormat.CommonSerieOptions.DropLines.LineColor = Color.Green;
    FileStream stream = new FileStream("DropLines.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
    workbook.Close();
}

```

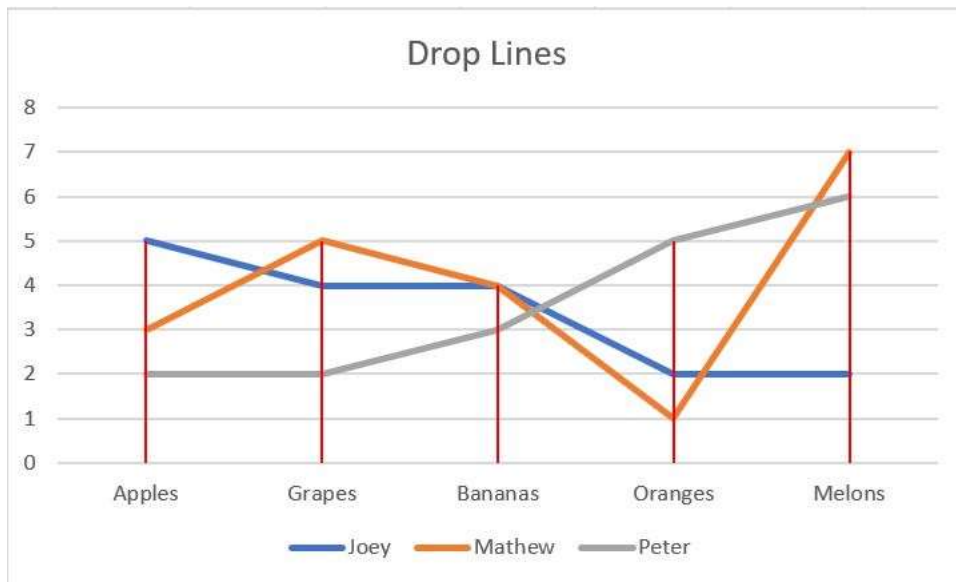
```
}

```

XAMARIN

```
using (ExcelEngine engine = new ExcelEngine())
{
    IApplication application = engine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    ///"App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    IChartShape chart = worksheet.Charts[0];
    IChartSerie chartSerie = chart.Series[0];
    ///Set HasDropLines property to true.
    chartSerie.SerieFormat.CommonSerieOptions.HasDropLines = true;
    ///Apply formats to DropLines.
    chartSerie.SerieFormat.CommonSerieOptions.DropLines.LineColor = Color.Green;
    ///Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    ///Save the document as file and view the saved document
    ///The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Refer to the xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("DropLines.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("DropLines.xlsx", "application/msexcel", stream);
    }
    workbook.Close();
}
```

The following screen shot shows the drop lines in the line chart.



Add Series Lines

Series lines are used in Excel stacked bar and column charts that create lines from one bar to another that connect every data point in a series.

Series lines in Excel Pie-of-pie and bar-of-pie charts are used to create lines that connect the main pie chart with the secondary pie or bar chart.

The following code snippet shows how to add series lines in a pie chart.

C#

```
using (ExcelEngine engine = new ExcelEngine())
{
    IApplication application = engine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    IChartShape chart = worksheet.Charts[0];
    IChartSerie chartSerie = chart.Series[0];
    //Set HasSeriesLines property to true.
    chartSerie.SerieFormat.CommonSerieOptions.HasSeriesLines = true;
    //Apply formats to SeriesLines.
    chartSerie.SerieFormat.CommonSerieOptions.PieSeriesLine.LineColor =
    Color.Red;
    workbook.SaveAs("SeriesLines.xlsx");
    workbook.Close();
}
```

VB.NET

```
Using engine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = engine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim chart As IChartShape = worksheet.Charts(0)
Dim chartSerie As IChartSerie = chart.Series(0)
'Set HasSeriesLines property to true.
chartSerie.SerieFormat.CommonSerieOptions.HasSeriesLines = True;
```

```
'Apply formats to SeriesLines.
chartSerie.SerieFormat.CommonSerieOptions.PieSeriesLine.LineColor =
Color.Red
workbook.SaveAs("SeriesLines.xlsx")
workbook.Close()
End Using
```

UWP

```
using (ExcelEngine engine = new ExcelEngine())
{
    IApplication application = engine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the file picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opening an existing workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    IChartShape chart = worksheet.Charts[0];
    IChartSerie chartSerie = chart.Series[0];
    //Set HasSeriesLines property to true.
    chartSerie.SerieFormat.CommonSerieOptions.HasSeriesLines = true;
    //Apply formats to SeriesLines.
    chartSerie.SerieFormat.CommonSerieOptions.PieSeriesLine.LineColor =
    Color.Red;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "SeriesLines";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine engine = new ExcelEngine())
{
    IApplication application = engine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    IChartShape chart = worksheet.Charts[0];
    IChartSerie chartSerie = chart.Series[0];
    //Set HasSeriesLines property to true.
    chartSerie.SerieFormat.CommonSerieOptions.HasSeriesLines = true;
    //Apply formats to SeriesLines.
```

```

chartSerie.SerieFormat.CommonSerieOptions.PieSeriesLine.LineColor =
Color.Red;
FileStream stream = new FileStream("SeriesLines.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
workbook.Close();
}

```

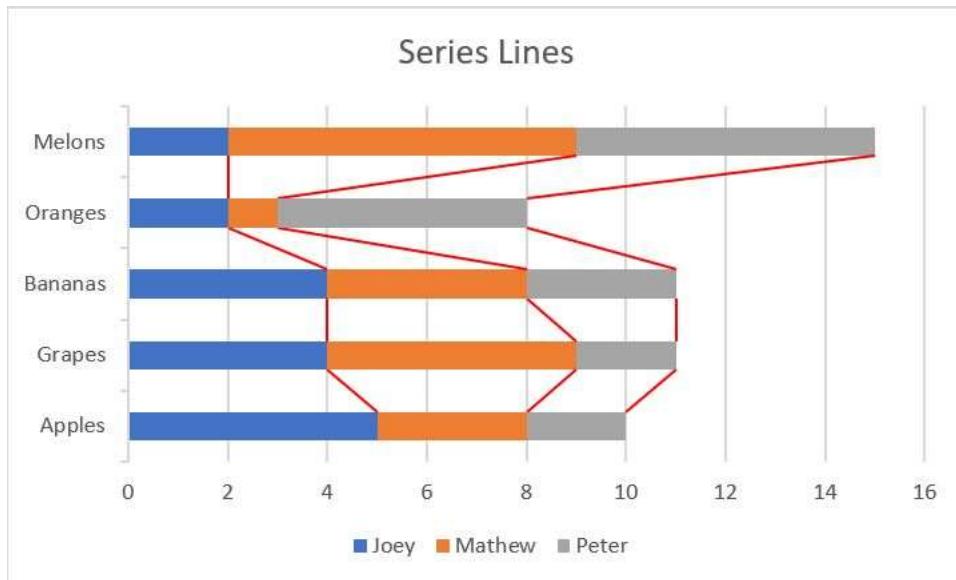
XAMARIN

```

using (ExcelEngine engine = new ExcelEngine())
{
    IApplication application = engine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    // "App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    IChartShape chart = worksheet.Charts[0];
    IChartSerie chartSerie = chart.Series[0];
    // Set HasSeriesLines property to true.
    chartSerie.SerieFormat.CommonSerieOptions.HasSeriesLines = true;
    // Apply formats to SeriesLines.
    chartSerie.SerieFormat.CommonSerieOptions.PieSeriesLine.LineColor =
Color.Red;
    // Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    // Save the document as file and view the saved document
    // The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Refer to the xlsio/xamarin section for respective
code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Series
Lines.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("SeriesLines.xlsx",
"application/msexcel", stream);
    }
    workbook.Close();
}

```

The following screen shot shows the series lines in the stacker bar chart.



Fill Chart Elements with Picture

Chart elements helps in modifying the chart appearance. The different chart elements are plot area, chart area, axes, titles, data points, legend, and data labels.

Fill plot area with picture

Plot area holds the data series of a chart. This plot area can be filled with solid colors, texture, picture, and pattern.

Essential XlsIO allows you to fill plot area with picture using the [UserPicture](#) of **IFill** interface. Refer to the following complete code snippets.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:C6"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Filling plot area of the chart with picture
    chart.PlotArea.Fill.UserPicture("Image.png");
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
```

```

'Adding chart in the worksheet
Dim chart As IChartShape = worksheet.Charts.Add
chart.DataRange = worksheet.Range("A1:C6")
chart.ChartType = ExcelChartType.Column_Clustered
chart.IsSeriesInRows = False
'Filling plot area of the chart with picture
chart.PlotArea.Fill.UserPicture("Image.png")
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the file picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opening an existing workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:C6"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //"App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream imageStream =
    assembly.GetManifestResourceStream("UWP.Data.Image.png");
    //Getting an image from the stream
    Syncfusion.XlsIO.Image image =
    Syncfusion.XlsIO.Image.FromStream(imageStream);
    //Filling plot area of the chart with picture
    chart.PlotArea.Fill.UserPicture(image, "Image");
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;

```

```

application.DefaultVersion = ExcelVersion.Excel2013;
FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream);
IWorksheet worksheet = workbook.Worksheets[0];
//Adding chart in the worksheet
IChartShape chart = worksheet.Charts.Add();
chart.DataRange = worksheet.Range["A1:C6"];
chart.ChartType = ExcelChartType.Column_Clustered;
chart.IsSeriesInRows = false;
//Getting an image from the stream
FileStream imageStream = new FileStream("Image.png", FileMode.Open,
    FileAccess.Read);
Image image = Image.FromStream(imageStream);
//Filling plot area of the chart with picture
chart.PlotArea.Fill.UserPicture(image, "Image");
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    ///"App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    ///Adding chart in the worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:C6"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    ///Getting an image from the stream
    Stream imageStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Image.png");
    Syncfusion.Drawing.Image image =
        Syncfusion.Drawing.Image.FromStream(imageStream);
    ///Filling plot area of the chart with picture
    chart.PlotArea.Fill.UserPicture(image, "Image");
    ///Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    ///Save the document as file and view the saved document
}

```



```
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android, and iOS platforms. Refer to the xlsio/xamarin section for
//respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}
```

Fill chart area with picture

Chart area holds plot area, legend, axes, data table, and so on. This chart area can be filled with solid colors, texture, picture, and pattern.

Similar to plot area, chart area can be filled with picture using [UserPicture](#) of [IFill](#) interface. Refer to the following complete code snippets.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
//Adding chart in the worksheet
IChartShape chart = worksheet.Charts.Add();
chart.DataRange = worksheet.Range["A1:C6"];
chart.ChartType = ExcelChartType.Column_Clustered;
chart.IsSeriesInRows = false;
//Filling chart area of the chart with picture
chart.ChartArea.Fill.UserPicture("Image.png");
workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding chart in the worksheet
Dim chart As IChartShape = worksheet.Charts.Add
chart.DataRange = worksheet.Range("A1:C6")
chart.ChartType = ExcelChartType.Column_Clustered
chart.IsSeriesInRows = False
'Filling chart area of the chart with picture
chart.ChartArea.Fill.UserPicture("Image.png")
workbook.SaveAs("Output.xlsx")
```

End Using**UWP**

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the file picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opening an existing workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:C6"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //"App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream imageStream =
    assembly.GetManifestResourceStream("UWP.Data.Image.png");
    //Getting an image from the stream
    Syncfusion.XlsIO.Image image =
    Syncfusion.XlsIO.Image.FromStream(imageStream);
    //Filling chart area of the chart with picture
    chart.ChartArea.Fill.UserPicture(image, "Image");
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:C6"];
}

```

```

chart.ChartType = ExcelChartType.Column_Clustered;
chart.IsSeriesInRows = false;
//Getting an image from the stream
FileStream imageStream = new FileStream("Image.png", FileMode.Open,
FileAccess.Read);
Image image = Image.FromStream(imageStream);
//Filling chart area of the chart with picture
chart.ChartArea.Fill.UserPicture(image, "Image");
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:C6"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Getting an image from the stream
    Stream imageStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Image.png");
    Syncfusion.Drawing.Image image =
    Syncfusion.Drawing.Image.FromStream(imageStream);
    //Filling chart area of the chart with picture
    chart.ChartArea.Fill.UserPicture(image, "Image");
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output.xlsx", "application/msexcel", stream);
    }
    else

```

```
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
    "application/msexcel", stream);
}
}
```

Applying 3D Formats

The following code example explains how to apply 3D settings such as rotation, side wall, back wall, and floor settings.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    IWorksheet sheet = workbook.Worksheets[0];
    //Insert the data in sheet-1
    sheet.Range["B1"].Text = "Product-A";
    sheet.Range["C1"].Text = "Product-B";
    sheet.Range["D1"].Text = "Product-C";
    sheet.Range["A2"].Text = "Jan";
    sheet.Range["A3"].Text = "Feb";
    sheet.Range["B2"].Number = 25;
    sheet.Range["B3"].Number = 20;
    sheet.Range["C2"].Number = 35;
    sheet.Range["C3"].Number = 25;
    sheet.Range["D2"].Number = 40;
    sheet.Range["D3"].Number = 55;
    IChartShape chart = sheet.Charts.Add();
    chart.DataRange = sheet.Range["A1:D3"];
    chart.ChartType = ExcelChartType.Column_Clustered_3D;
    //Set Rotation of the 3D chart view
    chart.Rotation = 90;
    //Set Back wall fill option
    chart.BackWall.Fill.FillType = ExcelFillType.Gradient;
    //Set Back wall thickness
    chart.BackWall.Thickness = 10;
    //Set Texture Type
    chart.BackWall.Fill.GradientColorType = ExcelGradientColor.TwoColor;
    chart.BackWall.Fill.GradientStyle = ExcelGradientStyle.Diagonal_Down;
    chart.BackWall.Fill.ForeColor = Color.WhiteSmoke;
    chart.BackWall.Fill.BackColor = Color.LightBlue;
    //Set side wall fill option
    chart.SideWall.Fill.FillType = ExcelFillType.SolidColor;
    //Set side wall fore and back color
    chart.SideWall.Fill.BackColor = Color.White;
    chart.SideWall.Fill.ForeColor = Color.White;
    //Set floor fill option
    chart.Floor.Fill.FillType = ExcelFillType.Pattern;
    chart.Floor.Fill.Pattern =
    ExcelGradientPattern.Pat_10_Percent.Pat_30_Percent;
    //Set floor fore and Back color
    chart.Floor.Fill.ForeColor = Color.Blue;
    chart.Floor.Fill.BackColor = Color.White;
}
```

```
//Set floor thickness
chart.Floor.Thickness = 3;
workbook.SaveAs ("Chart.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(2)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Insert data in sheet-1
sheet.Range("B1").Text = "Product-A"
sheet.Range("C1").Text = "Product-B"
sheet.Range("D1").Text = "Product-C"
sheet.Range("A2").Text = "Jan"
sheet.Range("A3").Text = "Feb"
sheet.Range("B2").Number = 25
sheet.Range("B3").Number = 20
sheet.Range("C2").Number = 35
sheet.Range("C3").Number = 25
sheet.Range("D2").Number = 40
sheet.Range("D3").Number = 55
Dim chart As IChartShape = sheet.Charts.Add()
chart.DataRange = sheet.Range("A1:D3")
chart.ChartType = ExcelChartType.Column_Clustered_3D
'Set Rotation of the 3D chart view
chart.Rotation = 90
'Set Back wall fill option
chart.BackWall.Fill.FillType = ExcelFillType.Gradient
'Set Texture Type
chart.BackWall.Fill.GradientColorType = ExcelGradientColor.TwoColor
chart.BackWall.Fill.GradientStyle = ExcelGradientStyle.Diagonal_Down
chart.BackWall.Fill.ForeColor = Color.WhiteSmoke
chart.BackWall.Fill.BackColor = Color.LightBlue
'Set Back wall thickness
chart.BackWall.Thickness = 10
'Set side wall fill option
chart.SideWall.Fill.FillType = ExcelFillType.SolidColor
'Set sidewall fore and back color
chart.SideWall.Fill.BackColor = Color.White
chart.SideWall.Fill.ForeColor = Color.White
'Set floor fill option
chart.Floor.Fill.FillType = ExcelFillType.Pattern
chart.Floor.Fill.Pattern =
ExcelGradientPattern.Pat_10_Percent.Pat_30_Percent
'Set floor fore and Back color
chart.Floor.Fill.ForeColor = Color.Blue
chart.Floor.Fill.BackColor = Color.White
'Set floor thickness
chart.Floor.Thickness = 3
workbook.SaveAs ("Chart.xlsx")
End Using
```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    IWorksheet sheet = workbook.Worksheets[0];
    //Insert the data in sheet-1
    sheet.Range["B1"].Text = "Product-A";
    sheet.Range["C1"].Text = "Product-B";
    sheet.Range["D1"].Text = "Product-C";
    sheet.Range["A2"].Text = "Jan";
    sheet.Range["A3"].Text = "Feb";
    sheet.Range["B2"].Number = 25;
    sheet.Range["B3"].Number = 20;
    sheet.Range["C2"].Number = 35;
    sheet.Range["C3"].Number = 25;
    sheet.Range["D2"].Number = 40;
    sheet.Range["D3"].Number = 55;
    IChartShape chart = sheet.Charts.Add();
    chart.DataRange = sheet.Range["A1:D3"];
    chart.ChartType = ExcelChartType.Column_Clustered_3D;
    //Set Rotation of the 3D chart view
    chart.Rotation = 90;
    //Set Back wall fill option
    chart.BackWall.Fill.FillType = ExcelFillType.Gradient;
    //Set Back wall thickness
    chart.BackWall.Thickness = 10;
    //Set Texture Type
    chart.BackWall.Fill.GradientColorType = ExcelGradientColor.TwoColor;
    chart.BackWall.Fill.GradientStyle = ExcelGradientStyle.Diagonal_Down;
    chart.BackWall.Fill.ForeColor = Color.FromArgb(255, 245, 245, 245);
    chart.BackWall.Fill.BackColor = Color.FromArgb(255, 173, 216, 230);
    //Set side wall fill option
    chart.SideWall.Fill.FillType = ExcelFillType.SolidColor;
    //Set side wall fore and back color
    chart.SideWall.Fill.BackColor = Color.FromArgb(255, 255, 255, 255);
    chart.SideWall.Fill.ForeColor = Color.FromArgb(255, 255, 255, 255);
    //Set floor fill option
    chart.Floor.Fill.FillType = ExcelFillType.Pattern;
    chart.Floor.Fill.Pattern =
    ExcelGradientPattern.Pat_10_Percent.Pat_30_Percent;
    //Set floor fore and Back color
    chart.Floor.Fill.ForeColor = Color.FromArgb(255, 0, 0, 255);
    chart.Floor.Fill.BackColor = Color.FromArgb(255, 255, 255, 255);
    //Set floor thickness
    chart.Floor.Thickness = 3;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Chart";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file

```

```
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    IWorksheet sheet = workbook.Worksheets[0];
    //Insert the data in sheet-1
    sheet.Range["B1"].Text = "Product-A";
    sheet.Range["C1"].Text = "Product-B";
    sheet.Range["D1"].Text = "Product-C";
    sheet.Range["A2"].Text = "Jan";
    sheet.Range["A3"].Text = "Feb";
    sheet.Range["B2"].Number = 25;
    sheet.Range["B3"].Number = 20;
    sheet.Range["C2"].Number = 35;
    sheet.Range["C3"].Number = 25;
    sheet.Range["D2"].Number = 40;
    sheet.Range["D3"].Number = 55;
    IChartShape chart = sheet.Charts.Add();
    chart.DataRange = sheet.Range["A1:D3"];
    chart.ChartType = ExcelChartType.Column_Clustered_3D;
    //Set Rotation of the 3D chart view
    chart.Rotation = 90;
    //Set Back wall fill option
    chart.BackWall.Fill.FillType = ExcelFillType.Gradient;
    //Set Back wall thickness
    chart.BackWall.Thickness = 10;
    //Set Texture Type
    chart.BackWall.Fill.GradientColorType = ExcelGradientColor.TwoColor;
    chart.BackWall.Fill.GradientStyle = ExcelGradientStyle.Diagonal_Down;
    chart.BackWall.Fill.ForeColor = Color.WhiteSmoke;
    chart.BackWall.Fill.BackColor = Color.LightBlue;
    //Set side wall fill option
    chart.SideWall.Fill.FillType = ExcelFillType.SolidColor;
    //Set side wall fore and back color
    chart.SideWall.Fill.BackColor = Color.White;
    chart.SideWall.Fill.ForeColor = Color.White;
    //Set floor fill option
    chart.Floor.Fill.FillType = ExcelFillType.Pattern;
    chart.Floor.Fill.Pattern =
    ExcelGradientPattern.Pat_10_Percent.Pat_30_Percent;
    //Set floor fore and Back color
    chart.Floor.Fill.ForeColor = Color.Blue;
    chart.Floor.Fill.BackColor = Color.White;
    //Set floor thickness
    chart.Floor.Thickness = 3;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Chart.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

```
}

```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(2);
    IWorksheet sheet = workbook.Worksheets[0];
    //Insert the data in sheet-1
    sheet.Range["B1"].Text = "Product-A";
    sheet.Range["C1"].Text = "Product-B";
    sheet.Range["D1"].Text = "Product-C";
    sheet.Range["A2"].Text = "Jan";
    sheet.Range["A3"].Text = "Feb";
    sheet.Range["B2"].Number = 25;
    sheet.Range["B3"].Number = 20;
    sheet.Range["C2"].Number = 35;
    sheet.Range["C3"].Number = 25;
    sheet.Range["D2"].Number = 40;
    sheet.Range["D3"].Number = 55;
    IChartShape chart = sheet.Charts.Add();
    chart.DataRange = sheet.Range["A1:D3"];
    chart.ChartType = ExcelChartType.Column_Clustered_3D;
    //Set Rotation of the 3D chart view
    chart.Rotation = 90;
    //Set Back wall fill option
    chart.BackWall.Fill.FillType = ExcelFillType.Gradient;
    //Set Back wall thickness
    chart.BackWall.Thickness = 10;
    //Set Texture Type
    chart.BackWall.Fill.GradientColorType = ExcelGradientColor.TwoColor;
    chart.BackWall.Fill.GradientStyle = ExcelGradientStyle.Diagonal_Down;
    chart.BackWall.Fill.ForeColor = Syncfusion.Drawing.Color.WhiteSmoke;
    chart.BackWall.Fill.BackColor = Syncfusion.Drawing.Color.LightBlue;
    //Set side wall fill option
    chart.SideWall.Fill.FillType = ExcelFillType.SolidColor;
    //Set side wall fore and back color
    chart.SideWall.Fill.BackColor = Syncfusion.Drawing.Color.White;
    chart.SideWall.Fill.ForeColor = Syncfusion.Drawing.Color.White;
    //Set floor fill option
    chart.Floor.Fill.FillType = ExcelFillType.Pattern;
    chart.Floor.Fill.Pattern =
    ExcelGradientPattern.Pat_10_Percent.Pat_30_Percent
    //Set floor fore and Back color
    chart.Floor.Fill.ForeColor = Syncfusion.Drawing.Color.Blue;
    chart.Floor.Fill.BackColor = Syncfusion.Drawing.Color.White;
    //Set floor thickness
    chart.Floor.Thickness = 3;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document

```



```
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Chart.
xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Chart.xlsx",
"application/msexcel", stream);
}
}
```

Customizing chart and Chart Elements

Positioning Chart

Chart can be positioned by specifying row and column indexes. The following code samples illustrates how to position a chart in a worksheet.

C#

```
//Positioning chart in a worksheet
chart.TopRow = 5;
chart.LeftColumn = 5;
chart.RightColumn = 10;
chart.BottomRow = 10;
```

VB.NET

```
'Positioning chart in a worksheet
chart.TopRow = 5
chart.LeftColumn = 5
chart.RightColumn = 10
chart.BottomRow = 10
```

UWP

```
//Positioning chart in a worksheet
chart.TopRow = 5;
chart.LeftColumn = 5;
chart.RightColumn = 10;
chart.BottomRow = 10;
```

ASP.NET CORE

```
//Positioning chart in a worksheet
chart.TopRow = 5;
chart.LeftColumn = 5;
chart.RightColumn = 10;
chart.BottomRow = 10;
```

XAMARIN

```
//Positioning chart in a worksheet
chart.TopRow = 5;
chart.LeftColumn = 5;
chart.RightColumn = 10;
chart.BottomRow = 10;
```

Positioning Chart Elements

The following code examples illustrate how to position the chart elements.

C#

```
//Manually positioning plot area
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.inner;
chart.PlotArea.Layout.LeftMode = LayoutModes.edge;
chart.PlotArea.Layout.TopMode = LayoutModes.edge;
//Manually positioning chart legend
chart.Legend.Layout.LeftMode = LayoutModes.edge;
chart.Legend.Layout.TopMode = LayoutModes.edge;
```

VB.NET

```
'Manually positioning plot area
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.inner
chart.PlotArea.Layout.LeftMode = LayoutModes.edge
chart.PlotArea.Layout.TopMode = LayoutModes.edge
'Manually positioning chart legend
chart.Legend.Layout.LeftMode = LayoutModes.edge
chart.Legend.Layout.TopMode = LayoutModes.edge
```

UWP

```
//Manually positioning plot area
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.inner;
chart.PlotArea.Layout.LeftMode = LayoutModes.edge;
chart.PlotArea.Layout.TopMode = LayoutModes.edge;
//Manually positioning chart legend
chart.Legend.Layout.LeftMode = LayoutModes.edge;
chart.Legend.Layout.TopMode = LayoutModes.edge;
```

ASP.NET CORE

```
//Manually positioning plot area
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.inner;
chart.PlotArea.Layout.LeftMode = LayoutModes.edge;
chart.PlotArea.Layout.TopMode = LayoutModes.edge;
//Manually positioning chart legend
chart.Legend.Layout.LeftMode = LayoutModes.edge;
chart.Legend.Layout.TopMode = LayoutModes.edge;
```

XAMARIN

```
//Manually positioning plot area
```

```
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.inner;
chart.PlotArea.Layout.LeftMode = LayoutModes.edge;
chart.PlotArea.Layout.TopMode = LayoutModes.edge;
//Manually positioning chart legend
chart.Legend.Layout.LeftMode = LayoutModes.edge;
chart.Legend.Layout.TopMode = LayoutModes.edge;
```

Resizing Chart

The following code sample illustrates how to resize a chart in a worksheet.

C#

```
IShape chartShape = chart as IShape;
//Set Height of the chart in pixels
chartShape.Height = 300;
//Set Width of the chart
chartShape.Width = 500;
```

VB.NET

```
Dim chartShape As IShape = chart as IShape
'Set Height of the chart
chartShape.Height = 300
'Set Width of the chart
chartShape.Width = 500
```

UWP

```
IShape chartShape = chart as IShape;
//Set Height of the chart in pixels
chartShape.Height = 300;
//Set Width of the chart
chartShape.Width = 500;
```

ASP.NET CORE

```
IShape chartShape = chart as IShape;
//Set Height of the chart in pixels
chartShape.Height = 300;
//Set Width of the chart
chartShape.Width = 500;
```

XAMARIN

```
IShape chartShape = chart as IShape;
//Set Height of the chart in pixels
chartShape.Height = 300;
//Set Width of the chart
chartShape.Width = 500;
```

Resizing Chart Elements

The following code examples illustrate how to resize chart elements such as plot area and legend.

C#

```
//Manually resizing chart plot area
chart.PlotArea.Layout.Left = 50;
chart.PlotArea.Layout.Top = 75;
chart.PlotArea.Layout.Width = 300;
chart.PlotArea.Layout.Height = 200;
//Manually resizing chart legend
chart.Legend.Layout.Left = 400;
chart.Legend.Layout.Top = 150;
chart.Legend.Layout.Width = 150;
chart.Legend.Layout.Height = 100;
```

VB.NET

```
'Manually resizing chart plot area
chart.PlotArea.Layout.Left = 50
chart.PlotArea.Layout.Top = 75
chart.PlotArea.Layout.Width = 300
chart.PlotArea.Layout.Height = 200
'Manually resizing chart legend
chart.Legend.Layout.Left = 400
chart.Legend.Layout.Top = 150
chart.Legend.Layout.Width = 150
chart.Legend.Layout.Height = 100
```

UWP

```
//Manually resizing chart plot area
chart.PlotArea.Layout.Left = 50;
chart.PlotArea.Layout.Top = 75;
chart.PlotArea.Layout.Width = 300;
chart.PlotArea.Layout.Height = 200;
//Manually resizing chart legend
chart.Legend.Layout.Left = 400;
chart.Legend.Layout.Top = 150;
chart.Legend.Layout.Width = 150;
chart.Legend.Layout.Height = 100;
```

ASP.NET CORE

```
//Manually resizing chart plot area
chart.PlotArea.Layout.Left = 50;
chart.PlotArea.Layout.Top = 75;
chart.PlotArea.Layout.Width = 300;
chart.PlotArea.Layout.Height = 200;
//Manually resizing chart legend
chart.Legend.Layout.Left = 400;
chart.Legend.Layout.Top = 150;
chart.Legend.Layout.Width = 150;
chart.Legend.Layout.Height = 100;
```

XAMARIN

```
//Manually resizing chart plot area
```

```

chart.PlotArea.Layout.Left = 50;
chart.PlotArea.Layout.Top = 75;
chart.PlotArea.Layout.Width = 300;
chart.PlotArea.Layout.Height = 200;
//Manually resizing chart legend
chart.Legend.Layout.Left = 400;
chart.Legend.Layout.Top = 150;
chart.Legend.Layout.Width = 150;
chart.Legend.Layout.Height = 100;

```

Chart with transparent background

The following code example explains how to apply transparency to chart area.

C#

```

//Applying transparency to chart area
chart.ChartArea.Fill.Transparency = 0.9;

```

VB.NET

```

'Applying transparency to chart area
chart.ChartArea.Fill.Transparency = 0.9

```

UWP

```

//Applying transparency to chart area
chart.ChartArea.Fill.Transparency = 0.9;

```

ASP.NET CORE

```

//Applying transparency to chart area
chart.ChartArea.Fill.Transparency = 0.9;

```

XAMARIN

```

//Applying transparency to chart area
chart.ChartArea.Fill.Transparency = 0.9;

```

The complete code snippet illustrating the above options is shown below.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    IChartShape chart = sheet.Charts[0];
    //Positioning chart in a worksheet
    chart.TopRow = 5;
    chart.LeftColumn = 5;
    chart.RightColumn = 10;
}

```

```

chart.BottomRow = 10;
//Manually positioning plot area
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.inner;
chart.PlotArea.Layout.LeftMode = LayoutModes.edge;
chart.PlotArea.Layout.TopMode = LayoutModes.edge;
//Manually positioning chart legend
chart.Legend.Layout.LeftMode = LayoutModes.edge;
chart.Legend.Layout.TopMode = LayoutModes.edge;
IShape chartShape = chart as IShape;
//Set Height of the chart in pixels
chartShape.Height = 300;
//Set Width of the chart
chartShape.Width = 500;
//Manually resizing chart plot area
chart.PlotArea.Layout.Left = 50;
chart.PlotArea.Layout.Top = 75;
chart.PlotArea.Layout.Width = 300;
chart.PlotArea.Layout.Height = 200;
//Manually resizing chart legend
chart.Legend.Layout.Left = 400;
chart.Legend.Layout.Top = 150;
chart.Legend.Layout.Width = 200;
chart.Legend.Layout.Height = 100;
//Applying transparency to chart area
chart.ChartArea.Fill.Transparency = 0.9;
workbook.SaveAs ("Chart.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim chart As IChartShape = sheet.Charts(0)
'Positioning chart in a worksheet
chart.TopRow = 5
chart.LeftColumn = 5
chart.RightColumn = 10
chart.BottomRow = 10
'Manually positioning plot area
chart.PlotArea.Layout.LayoutTarget = LayoutTargets.inner
chart.PlotArea.Layout.LeftMode = LayoutModes.edge
chart.PlotArea.Layout.TopMode = LayoutModes.edge
'Manually positioning chart legend
chart.Legend.Layout.LeftMode = LayoutModes.edge
chart.Legend.Layout.TopMode = LayoutModes.edge
Dim chartShape As IShape = TryCast(chart, IShape)
'Set Height of the chart in pixels
chartShape.Height = 300
'Set Width of the chart
chartShape.Width = 500
'Manually resizing chart plot area
chart.PlotArea.Layout.Left = 50

```

```

chart.PlotArea.Layout.Top = 75
chart.PlotArea.Layout.Width = 300
chart.PlotArea.Layout.Height = 200
'Manually resizing chart legend'
chart.Legend.Layout.Left = 400
chart.Legend.Layout.Top = 150
chart.Legend.Layout.Width = 200
chart.Legend.Layout.Height = 100
'Applying transparency to chart area'
chart.ChartArea.Fill.Transparency = 0.9
workbook.SaveAs("Chart.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet sheet = workbook.Worksheets[0];
    IChartShape chart = sheet.Charts[0];
    //Positioning chart in a worksheet
    chart.TopRow = 5;
    chart.LeftColumn = 5;
    chart.RightColumn = 10;
    chart.BottomRow = 10;
    //Manually positioning plot area
    chart.PlotArea.Layout.LayoutTarget = LayoutTargets.inner;
    chart.PlotArea.Layout.LeftMode = LayoutModes.edge;
    chart.PlotArea.Layout.TopMode = LayoutModes.edge;
    //Manually positioning chart legend
    chart.Legend.Layout.LeftMode = LayoutModes.edge;
    chart.Legend.Layout.TopMode = LayoutModes.edge;
    IShape chartShape = chart as IShape;
    //Set Height of the chart in pixels
    chartShape.Height = 300;
    //Set Width of the chart
    chartShape.Width = 500;
    //Manually resizing chart plot area
    chart.PlotArea.Layout.Left = 50;
    chart.PlotArea.Layout.Top = 75;
    chart.PlotArea.Layout.Width = 300;
    chart.PlotArea.Layout.Height = 200;
    //Manually resizing chart legend
    chart.Legend.Layout.Left = 400;
    chart.Legend.Layout.Top = 150;
    chart.Legend.Layout.Width = 200;
    chart.Legend.Layout.Height = 100;
}

```

```

//Applying transparency to chart area
chart.ChartArea.Fill.Transparency = 0.9;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Chart";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    IChartShape chart = sheet.Charts[0];
    //Positioning chart in a worksheet
    chart.TopRow = 5;
    chart.LeftColumn = 5;
    chart.RightColumn = 10;
    chart.BottomRow = 10;
    //Manually positioning plot area
    chart.PlotArea.Layout.LayoutTarget = LayoutTargets.inner;
    chart.PlotArea.Layout.LeftMode = LayoutModes.edge;
    chart.PlotArea.Layout.TopMode = LayoutModes.edge;
    //Manually positioning chart legend
    chart.Legend.Layout.LeftMode = LayoutModes.edge;
    chart.Legend.Layout.TopMode = LayoutModes.edge;
    IShape chartShape = chart as IShape;
    //Set Height of the chart in pixels
    chartShape.Height = 300;
    //Set Width of the chart
    chartShape.Width = 500;
    //Manually resizing chart plot area
    chart.PlotArea.Layout.Left = 50;
    chart.PlotArea.Layout.Top = 75;
    chart.PlotArea.Layout.Width = 300;
    chart.PlotArea.Layout.Height = 200;
    //Manually resizing chart legend
    chart.Legend.Layout.Left = 400;
    chart.Legend.Layout.Top = 150;
    chart.Legend.Layout.Width = 200;
    chart.Legend.Layout.Height = 100;
    //Applying transparency to chart area
    chart.ChartArea.Fill.Transparency = 0.9;
    //Saving the workbook as stream
}

```



```

FileStream stream = new FileStream("Chart.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    IChartShape chart = sheet.Charts[0];
    ///Positioning chart in a worksheet
    chart.TopRow = 5;
    chart.LeftColumn = 5;
    chart.RightColumn = 10;
    chart.BottomRow = 10;
    ///Manually positioning plot area
    chart.PlotArea.Layout.LayoutTarget = LayoutTargets.inner;
    chart.PlotArea.Layout.LeftMode = LayoutModes.edge;
    chart.PlotArea.Layout.TopMode = LayoutModes.edge;
    ///Manually positioning chart legend
    chart.Legend.Layout.LeftMode = LayoutModes.edge;
    chart.Legend.Layout.TopMode = LayoutModes.edge;
    IShape chartShape = chart as IShape;
    ///Set Height of the chart in pixels
    chartShape.Height = 300;
    ///Set Width of the chart
    chartShape.Width = 500;
    ///Manually resizing chart plot area
    chart.PlotArea.Layout.Left = 50;
    chart.PlotArea.Layout.Top = 75;
    chart.PlotArea.Layout.Width = 300;
    chart.PlotArea.Layout.Height = 200;
    ///Manually resizing chart legend
    chart.Legend.Layout.Left = 400;
    chart.Legend.Layout.Top = 150;
    chart.Legend.Layout.Width = 200;
    chart.Legend.Layout.Height = 100;
    ///Applying transparency to chart area
    chart.ChartArea.Fill.Transparency = 0.9;
    ///Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    ///Save the document as file and view the saved document
}

```

```
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Chart.
xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Chart.xlsx",
"application/msexcel", stream);
}
}
```

Note: In order to position the chart elements, plot area should be smaller than chart area.

Explode a Pie Chart

Essential XlsIO allows you to explode either all data points at a single explosion value or each data point at different explosion using [Percent](#) of **ISeriesDataFormat** interface.

You can either create a pie chart and then explode it or directly create an exploded pie chart using XlsIO. Selecting **Pie_Exploded** as **ChartType** inserts a pie chart with a default explosion of **25%**. Learn how to [Create an Exploded Pie Chart](#).

Refer the following complete code snippets.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
//Adding pie chart in the worksheet
IChartShape chart = worksheet.Charts.Add();
chart.DataRange = worksheet.Range["A3:B7"];
chart.ChartType = ExcelChartType.Pie;
chart.IsSeriesInRows = false;
//Showing the values of data points
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
//Exploding the pie chart to 40%
chart.Series[0].SerieFormat.Percent = 40;
workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding pie chart in the worksheet
```

```

Dim chart As IChartShape = worksheet.Charts.Add
chart.DataRange = worksheet.Range("A3:B7")
chart.ChartType = ExcelChartType.Pie
chart.IsSeriesInRows = False
'Showing the values of data points
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.IsValue = True
'Exploding the pie chart to 40%
chart.Series(0).SerieFormat.Percent = 40
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the file picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opening an existing workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding pie chart in the worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A3:B7"];
    chart.ChartType = ExcelChartType.Pie;
    chart.IsSeriesInRows = false;
    //Showing the values of data points
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    //Exploding the pie chart to 40%
    chart.Series[0].SerieFormat.Percent = 40;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
}

```

```

IWorksheet worksheet = workbook.Worksheets[0];
//Adding pie chart in the worksheet
IChartShape chart = worksheet.Charts.Add();
chart.DataRange = worksheet.Range["A3:B7"];
chart.ChartType = ExcelChartType.Pie;
chart.IsSeriesInRows = false;
//Showing the values of data points
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
//Exploding the pie chart to 40%
chart.Series[0].SerieFormat.Percent = 40;
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    // "App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding pie chart in the worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A3:B7"];
    chart.ChartType = ExcelChartType.Pie;
    chart.IsSeriesInRows = false;
    //Showing the values of data points
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    //Exploding the pie chart to 40%
    chart.Series[0].SerieFormat.Percent = 40;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output.xlsx", "application/msexcel", stream);
    }
    else
    {

```

```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}
```

Add Picture to Chart and assign Hyperlink

Essential XlsIO supports assigning hyperlink to the picture added in a chart in the Excel workbook. To achieve this, create a [chart in workbook](#) and add picture to the chart using [AddPicture](#) of **IPictures** interface. You can assign hyperlink to the picture using [Add](#) property of **IHyperlinks** interface.

Refer to the following complete code snippets.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the workbook
    IChart chart = workbook.Charts.Add();
    chart.DataRange = worksheet.Range["A1:B6"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Adding picture on the chart
    chart.Pictures.AddPicture("Image.png");
    //Adding hyperlink to the picture on chart
    worksheet.HyperLinks.Add((workbook.Charts[0].Pictures[0] as IShape),
    ExcelHyperLinkType.Url, "http://www.Syncfusion.com", "click here");
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding chart in the workbook
Dim chart As IChart = workbook.Charts.Add
chart.DataRange = worksheet.Range("A1:B6")
chart.ChartType = ExcelChartType.Column_Clustered
chart.IsSeriesInRows = False
'Adding picture on the chart
chart.Pictures.AddPicture("Image.png")
'Adding hyperlink to the picture on chart
worksheet.HyperLinks.Add(workbook.Charts(0).Pictures(0),
ExcelHyperLinkType.Url, "http://www.Syncfusion.com", "click here")
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the file picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opening an existing workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the workbook
    IChart chart = workbook.Charts.Add();
    chart.DataRange = worksheet.Range["A1:B6"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    /"App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream imageStream =
    assembly.GetManifestResourceStream("UWP.Data.Image.png");
    //Adding picture on the chart
    chart.Pictures.AddPicture(1,1,imageStream);
    //Adding hyperlink to the picture on chart
    worksheet.HyperLinks.Add((workbook.Charts[0].Pictures[0] as IShape),
    ExcelHyperLinkType.Url, "http://www.Syncfusion.com", "click here");
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the workbook
    IChart chart = workbook.Charts.Add();
    chart.DataRange = worksheet.Range["A1:B6"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Getting an image from the stream

```

```

FileStream imageStream = new FileStream("Image.png", FileMode.Open,
    FileAccess.Read);
Image image = Image.FromStream(imageStream);
//Adding picture on the chart
chart.Pictures.AddPicture(1, 1, imageStream);
//Adding hyperlink to the picture on chart
worksheet.HyperLinks.Add((workbook.Charts[0].Pictures[0] as IShape),
    ExcelHyperLinkType.Url, "http://www.Syncfusion.com", "click here");
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding chart in the workbook
    IChart chart = workbook.Charts.Add();
    chart.DataRange = worksheet.Range["A1:B6"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Getting an image from the stream
    Stream imageStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Image.png");
    Syncfusion.Drawing.Image image =
        Syncfusion.Drawing.Image.FromStream(imageStream);
    //Adding picture on the chart
    chart.Pictures.AddPicture(1, 1, imageStream);
    //Adding hyperlink to the picture on chart
    worksheet.HyperLinks.Add((workbook.Charts[0].Pictures[0] as IShape),
        ExcelHyperLinkType.Url, "http://www.Syncfusion.com", "click here");
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {

```

```

Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Note: XlsIO supports adding picture only to a chart in the workbook, but does not support adding picture to a chart in the worksheet.

Add DataTable to Chart

Data table beneath the chart clearly represents the chart content in table format. While creating a chart, the data table is hidden, and the option should be manually enabled to view it.

Essential XlsIO supports adding data table using [HasDataTable](#) of **IChart** interface. Enabling this property adds the data table beneath the chart.

Refer the following complete code snippets.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Assigning data in the worksheet
    worksheet.Range["A1"].Text = "Items";
    worksheet.Range["B1"].Text = "Amount(in $)";
    worksheet.Range["C1"].Text = "Count";
    worksheet.Range["A2"].Text = "Beverages";
    worksheet.Range["A3"].Text = "Condiments";
    worksheet.Range["A4"].Text = "Confections";
    worksheet.Range["A5"].Text = "Dairy Products";
    worksheet.Range["A6"].Text = "Grains / Cereals";
    worksheet.Range["B2"].Number = 2776;
    worksheet.Range["B3"].Number = 1077;
    worksheet.Range["B4"].Number = 2287;
    worksheet.Range["B5"].Number = 1368;
    worksheet.Range["B6"].Number = 3325;
    worksheet.Range["C2"].Number = 925;
    worksheet.Range["C3"].Number = 378;
    worksheet.Range["C4"].Number = 880;
    worksheet.Range["C5"].Number = 581;
    worksheet.Range["C6"].Number = 189;
    //Adding a chart to the worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:C6"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Adding title to the chart
    chart.ChartTitle = "Chart with Data Table";
}

```



```
//Adding data table to the chart
chart.HasDataTable = true;
workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Assigning data in the worksheet
worksheet.Range("A1").Text = "Items"
worksheet.Range("B1").Text = "Amount(in $)"
worksheet.Range("C1").Text = "Count"
worksheet.Range("A2").Text = "Beverages"
worksheet.Range("A3").Text = "Condiments"
worksheet.Range("A4").Text = "Confections"
worksheet.Range("A5").Text = "Dairy Products"
worksheet.Range("A6").Text = "Grains / Cereals"
worksheet.Range("B2").Number = 2776
worksheet.Range("B3").Number = 1077
worksheet.Range("B4").Number = 2287
worksheet.Range("B5").Number = 1368
worksheet.Range("B6").Number = 3325
worksheet.Range("C2").Number = 925
worksheet.Range("C3").Number = 378
worksheet.Range("C4").Number = 880
worksheet.Range("C5").Number = 581
worksheet.Range("C6").Number = 189
'Adding a chart in the worksheet
Dim chart As IChartShape = worksheet.Charts.Add
chart.DataRange = worksheet.Range("A1:C6")
chart.ChartType = ExcelChartType.Column_Clustered
chart.IsSeriesInRows = False
'Adding title to the chart
chart.ChartTitle = "Chart with Data Table"
'Adding data table to the chart
chart.HasDataTable = True
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Assigning data in the worksheet
worksheet.Range["A1"].Text = "Items";
worksheet.Range["B1"].Text = "Amount(in $)";
worksheet.Range["C1"].Text = "Count";
worksheet.Range["A2"].Text = "Beverages";
```

```

worksheet.Range["A3"].Text = "Condiments";
worksheet.Range["A4"].Text = "Confections";
worksheet.Range["A5"].Text = "Dairy Products";
worksheet.Range["A6"].Text = "Grains / Cereals";
worksheet.Range["B2"].Number = 2776;
worksheet.Range["B3"].Number = 1077;
worksheet.Range["B4"].Number = 2287;
worksheet.Range["B5"].Number = 1368;
worksheet.Range["B6"].Number = 3325;
worksheet.Range["C2"].Number = 925;
worksheet.Range["C3"].Number = 378;
worksheet.Range["C4"].Number = 880;
worksheet.Range["C5"].Number = 581;
worksheet.Range["C6"].Number = 189;
//Adding a chart to the worksheet
IChartShape chart = worksheet.Charts.Add();
chart.DataRange = worksheet.Range["A1:C6"];
chart.ChartType = ExcelChartType.Column_Clustered;
chart.IsSeriesInRows = false;
//Adding title to the chart
chart.ChartTitle = "Chart with Data Table";
//Adding data table to the chart
chart.HasDataTable = true;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Assigning data in the worksheet
    worksheet.Range["A1"].Text = "Items";
    worksheet.Range["B1"].Text = "Amount(in $)";
    worksheet.Range["C1"].Text = "Count";
    worksheet.Range["A2"].Text = "Beverages";
    worksheet.Range["A3"].Text = "Condiments";
    worksheet.Range["A4"].Text = "Confections";
    worksheet.Range["A5"].Text = "Dairy Products";
    worksheet.Range["A6"].Text = "Grains / Cereals";
    worksheet.Range["B2"].Number = 2776;
    worksheet.Range["B3"].Number = 1077;
    worksheet.Range["B4"].Number = 2287;
    worksheet.Range["B5"].Number = 1368;
}

```

```

worksheet.Range["B6"].Number = 3325;
worksheet.Range["C2"].Number = 925;
worksheet.Range["C3"].Number = 378;
worksheet.Range["C4"].Number = 880;
worksheet.Range["C5"].Number = 581;
worksheet.Range["C6"].Number = 189;
//Adding a chart to the worksheet
IChartShape chart = worksheet.Charts.Add();
chart.DataRange = worksheet.Range["A1:C6"];
chart.ChartType = ExcelChartType.Column_Clustered;
chart.IsSeriesInRows = false;
//Adding title to the chart
chart.ChartTitle = "Chart with Data Table";
//Adding data table to the chart
chart.HasDataTable = true;
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Assigning data in the worksheet
    worksheet.Range["A1"].Text = "Items";
    worksheet.Range["B1"].Text = "Amount(in $)";
    worksheet.Range["C1"].Text = "Count";
    worksheet.Range["A2"].Text = "Beverages";
    worksheet.Range["A3"].Text = "Condiments";
    worksheet.Range["A4"].Text = "Confections";
    worksheet.Range["A5"].Text = "Dairy Products";
    worksheet.Range["A6"].Text = "Grains / Cereals";
    worksheet.Range["B2"].Number = 2776;
    worksheet.Range["B3"].Number = 1077;
    worksheet.Range["B4"].Number = 2287;
    worksheet.Range["B5"].Number = 1368;
    worksheet.Range["B6"].Number = 3325;
    worksheet.Range["C2"].Number = 925;
    worksheet.Range["C3"].Number = 378;
    worksheet.Range["C4"].Number = 880;
    worksheet.Range["C5"].Number = 581;
    worksheet.Range["C6"].Number = 189;
    //Adding chart to the worksheet
    IChartShape chart = worksheet.Charts.Add();
    chart.DataRange = worksheet.Range["A1:C6"];
    chart.ChartType = ExcelChartType.Column_Clustered;
    chart.IsSeriesInRows = false;
    //Adding title to the chart
    chart.ChartTitle = "Chart with Data Table";
}

```

```
//Adding data table to the chart
chart.HasDataTable = true;
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android, and iOS platforms. Refer to the xlsio/xamarin section for
//respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}
```

Sparkline

[Sparkline](#) is a small chart in a worksheet cell that provides a visual representation of data.

Sparkline Creation Using XlsIO

XlsIO provides support for creation, modification and removal of Sparklines.

- **ISparklineGroups** interface caches the SparklineGroup that need to be added to the Spreadsheet.
- **ISparklineGroup** represents Sparklines in object, and has properties that allows to customize it.
- **ISparklines** interface returns the collection of Sparkline present in a Worksheet.
- **ISparkline** represents a sparkline in the Sparklines. Currently, XlsIO supports all the three types of sparklines - Line, Column, Win/Loss.

Following code example illustrates how to create Sparklines by using XlsIO.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("spark.xlsx",
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
//Add SparklineGroups
ISparklineGroup sparklineGroup = sheet.SparklineGroups.Add();
//Add SparkLineType
sparklineGroup.SparklineType = SparklineType.Line;
sparklineGroup.MarkersColor = Color.BlueViolet;
//Add sparklines
ISparklines sparklines = sparklineGroup.Add();
```

```

IRange dataRange = sheet.Range["B2:F4"];
IRange referenceRange = sheet.Range["G2:G4"];
sparklines.Add(dataRange, referenceRange);
string fileName = "Sparkline.xlsx";
workbook.SaveAs(fileName);
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("spark.xlsx",
ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Add SparklineGroups
Dim sparklineGroup As ISparklineGroup = sheet.SparklineGroups.Add()
'Add SparkLineType
sparklineGroup.SparklineType = SparklineType.Line
sparklineGroup.MarkersColor = Color.BlueViolet
'Add sparklines
Dim sparklines As ISparklines = sparklineGroup.Add()
Dim dataRange As IRange = sheet.Range("B2:F4")
Dim referenceRange As IRange = sheet.Range("G2:G4")
sparklines.Add(dataRange, referenceRange)
Dim fileName As String = "Sparkline.xlsx"
workbook.SaveAs(fileName)
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
//Add SparklineGroups
ISparklineGroup sparklineGroup = sheet.SparklineGroups.Add();
//Add SparkLineType
sparklineGroup.SparklineType = SparklineType.Line;
sparklineGroup.MarkersColor = Color.FromArgb(255, 138, 43, 226);
//Add sparklines
ISparklines sparklines = sparklineGroup.Add();
IRange dataRange = sheet.Range["B2:F4"];
IRange referenceRange = sheet.Range["G2:G4"];
sparklines.Add(dataRange, referenceRange);
//Initializes FileSavePicker
}

```

```

FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sparkline";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("spark.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Add SparklineGroups
    ISparklineGroup sparklineGroup = sheet.SparklineGroups.Add();
    //Add SparkLineType
    sparklineGroup.SparklineType = SparklineType.Line;
    sparklineGroup.MarkersColor = Color.BlueViolet;
    //Add sparklines
    ISparklines sparklines = sparklineGroup.Add();
    IRange dataRange = sheet.Range["B2:F4"];
    IRange referenceRange = sheet.Range["G2:G4"];
    sparklines.Add(dataRange, referenceRange);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Sparkline.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.spa
    rk.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Add SparklineGroups
    ISparklineGroup sparklineGroup = sheet.SparklineGroups.Add();
    //Add SparkLineType

```

```

sparklineGroup.SparklineType = SparklineType.Line;
sparklineGroup.MarkersColor = Syncfusion.Drawing.Color.BlueViolet;
//Add sparklines
ISparklines sparklines = sparklineGroup.Add();
IRange dataRange = sheet.Range["B2:F4"];
IRange referenceRange = sheet.Range["G2:G4"];
sparklines.Add(dataRange, referenceRange);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Sparkl
ine.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sparkline.xlsx",
"application/msexcel", stream);
}
}

```

Modifying an existing spark line

XlsIO provides an option to edit the data of existing Sparklines. The following code snippet shows how to achieve this.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sparkline.xlsx",
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
ISparklineGroup sparklineGroup = sheet.SparklineGroups[0];
ISparklines sparklines = sparklineGroup[0];
IRange dataRange = sheet["A1:C4"];
IRange referenceRange = sheet["D1:D4"];
//Edit the existing sparklines data
sparklines.RefreshRanges(dataRange, referenceRange);
workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013

```

```

Dim workbook As IWorkbook = application.Workbooks.Open("Sparkline.xlsx",
ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim sparklineGroup As ISparklineGroup = sheet.SparklineGroups(0)
Dim sparklines As ISparklines = sparklineGroup(0)
Dim dataRange As IRange = sheet("A1:C4")
Dim referenceRange As IRange = sheet("D1:D4")
'Edit the existing sparklines data
sparklines.RefreshRanges(dataRange, referenceRange)
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    ISparklineGroup sparklineGroup = sheet.SparklineGroups[0];
    ISparklines sparklines = sparklineGroup[0];
    IRange dataRange = sheet["A1:C4"];
    IRange referenceRange = sheet["D1:D4"];
    //Edit the existing sparklines data
    sparklines.RefreshRanges(dataRange, referenceRange);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("Sparkline.xlsx", FileMode.Open,
FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
}

```



```

IWorksheet sheet = workbook.Worksheets[0];
ISparklineGroup sparklineGroup = sheet.SparklineGroups[0];
ISparklines sparklines = sparklineGroup[0];
IRange dataRange = sheet["A1:C4"];
IRange referenceRange = sheet["D1:D4"];
//Edit the existing sparklines data
sparklines.RefreshRanges(dataRange, referenceRange);
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    // "App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.spark.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    ISparklineGroup sparklineGroup = sheet.SparklineGroups[0];
    ISparklines sparklines = sparklineGroup[0];
    IRange dataRange = sheet["A1:C4"];
    IRange referenceRange = sheet["D1:D4"];
    //Edit the existing sparklines data
    sparklines.RefreshRanges(dataRange, referenceRange);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx", "application/msexcel", stream);
    }
}

```

Removing Sparklines

XlsIO provides an API to remove sparklines from the sparkline group and also the sparkline group from the worksheet. This is illustrated in the following code.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sparkline.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    ISparklineGroup sparklineGroup = sheet.SparklineGroups[0];
    ISparklines sparklines = sparklineGroup[0];
    //Remove sparkline specified by index from the sparklines
    sparklines.Remove(sparklines[1]);
    //Remove sparklines from the sparkline group
    sparklineGroup.Remove(sparklines);
    //Remove sparkline group from the sheet
    sheet.SparklineGroups.Remove(sparklineGroup);
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
    ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim sparklineGroup As ISparklineGroup = sheet.SparklineGroups(0)
Dim sparklines As ISparklines = sparklineGroup(0)
'Remove sparkline specified by index from the sparklines
sparklines.Remove(sparklines(1))
'Remove sparklines from the sparkline group
sparklineGroup.Remove(sparklines)
'Remove sparkline group from the sheet
sheet.SparklineGroups.Remove(sparklineGroup)
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
}
```

```

IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
ISparklineGroup sparklineGroup = sheet.SparklineGroups[0];
ISparklines sparklines = sparklineGroup[0];
//Remove sparkline specified by index from the sparklines
sparklines.Remove(sparklines[1]);
//Remove sparklines from the sparkline group
sparklineGroup.Remove(sparklines);
//Remove sparkline group from the sheet
sheet.SparklineGroups.Remove(sparklineGroup);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
FileStream inputStream = new FileStream("Sparkline.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
ISparklineGroup sparklineGroup = sheet.SparklineGroups[0];
ISparklines sparklines = sparklineGroup[0];
//Remove sparkline specified by index from the sparklines
sparklines.Remove(sparklines[1]);
//Remove sparklines from the sparkline group
sparklineGroup.Remove(sparklines);
//Remove sparkline group from the sheet
sheet.SparklineGroups.Remove(sparklineGroup);
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//"App" is the class of Portable project

```

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.spark.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
ISparklineGroup sparklineGroup = sheet.SparklineGroups[0];
ISparklines sparklines = sparklineGroup[0];
//Remove sparkline specified by index from the sparklines
sparklines.Remove(sparklines[1]);
//Remove sparklines from the sparkline group
sparklineGroup.Remove(sparklines);
//Remove sparkline group from the sheet
sheet.SparklineGroups.Remove(sparklineGroup);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Note: Sparklines are supported only from Excel 2007 onwards and are ignored in the earlier versions.

Excel 2016 Charts

Essential XlsIO supports creating and manipulating new and modern chart types such as waterfall, histogram, pareto, box and whisker, tree map, and sunburst, all of which are introduced in Microsoft Excel 2016.

Funnel

[Funnel](#) charts show values across multiple stages in a process.

Following code example illustrates how to create Funnel chart.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);

```

```

IWorksheet sheet = workbook.Worksheets[0];
//Create a chart
IChartShape chart = sheet.Charts.Add();
//Set chart type as Funnel
chart.ChartType = ExcelChartType.Funnel;
//Set data range in the worksheet
chart.DataRange = sheet.Range["A1:B6"];
//Set the chart title
chart.ChartTitle = "Funnel";
//Formatting the legend and data label option
chart.HasLegend = false;
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
workbook.SaveAs("Funnel.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Create a chart
Dim chart As IChartShape = sheet.Charts.Add()
'Set chart type as Funnel
chart.ChartType = ExcelChartType.Funnel
'Set data range in the worksheet
chart.DataRange = sheet.Range("A1:B6")
'Set the chart title
chart.ChartTitle = "Funnel"
'Formatting the legend and data label option
chart.HasLegend = False
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.IsValue = True
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.Size = 8
workbook.SaveAs("Funnel.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
//Create a chart

```

```

IChartShape chart = sheet.Charts.Add();
//Set chart type as Funnel
chart.ChartType = ExcelChartType.Funnel;
//Set data range in the worksheet
chart.DataRange = sheet.Range["A1:B6"];
//Set the chart title
chart.ChartTitle = "Funnel";
//Formatting the legend and data label option
chart.HasLegend = false;
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Funnel1";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as Funnel
    chart.ChartType = ExcelChartType.Funnel;
    //Set data range in the worksheet
    chart.DataRange = sheet.Range["A1:B6"];
    //Set the chart title
    chart.ChartTitle = "Funnel";
    //Formatting the legend and data label option
    chart.HasLegend = false;
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Funnel.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())

```

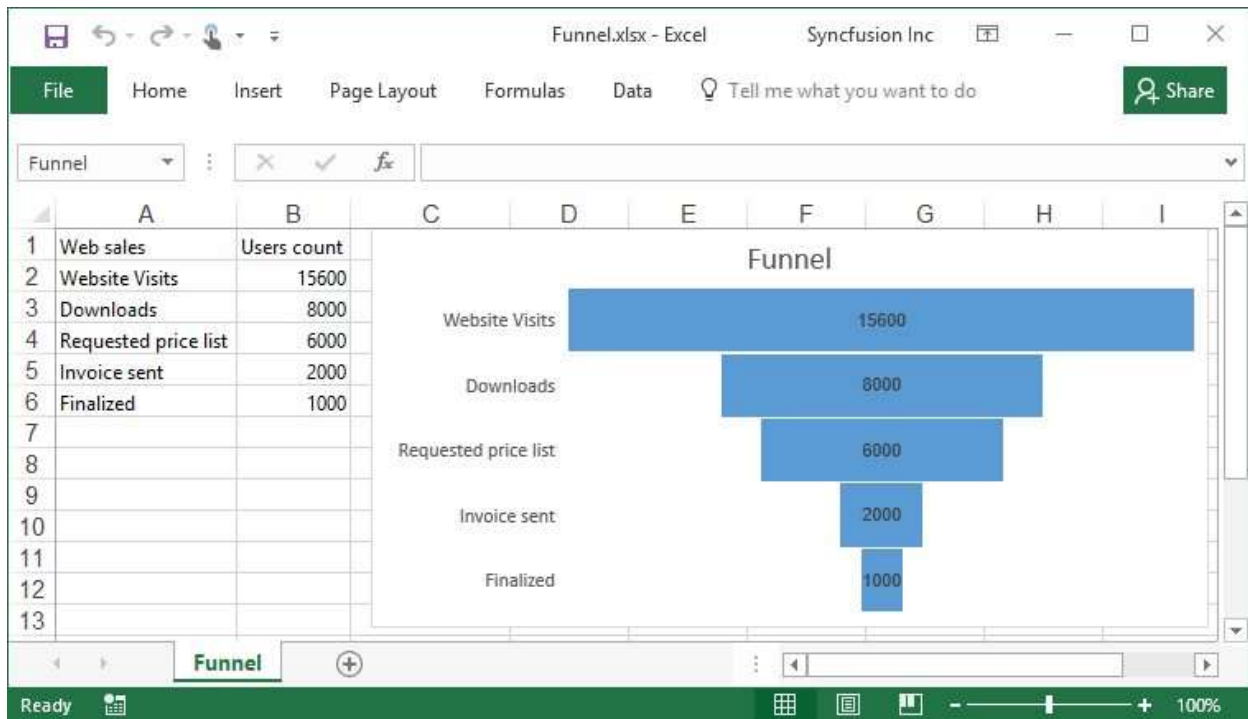
```

{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    // "App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    // Create a chart
    IChartShape chart = sheet.Charts.Add();
    // Set chart type as Funnel
    chart.ChartType = ExcelChartType.Funnel;
    // Set data range in the worksheet
    chart.DataRange = sheet.Range["A1:B6"];
    // Set the chart title
    chart.ChartTitle = "Funnel";
    // Formatting the legend and data label option
    chart.HasLegend = false;
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
    // Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    // Save the document as file and view the saved document
    // The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Funnel.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Funnel.xlsx", "application/msexcel", stream);
    }
}

```

The input template can be downloaded [here](#).

The following screen shot shows the output of above code.



Box and Whisker

[Box and Whisker](#) chart shows distribution of data into quartiles, highlighting the mean and outliers. Box and Whisker charts are most commonly used in statistical analysis.

Following code example illustrates how to create Box and Whisker chart.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set the chart title
    chart.ChartTitle = "Test Scores";
    //Set chart type as Box and Whisker
    chart.ChartType = ExcelChartType.BoxAndWhisker;
    //Set data range in the worksheet
    chart.DataRange = sheet["A1:D16"];
    //Box and Whisker settings on first series
    IChartSeries seriesA = chart.Series[0];
    seriesA.SeriesFormat.ShowInnerPoints = false;
    seriesA.SeriesFormat.ShowOutlierPoints = true;
    seriesA.SeriesFormat.ShowMeanMarkers = true;
    seriesA.SeriesFormat.ShowMeanLine = false;
    seriesA.SeriesFormat.QuartileCalculationType =
        ExcelQuartileCalculation.ExclusiveMedian;
    //Box and Whisker settings on second series
    IChartSeries seriesB = chart.Series[1];
}
```



```

seriesB.SerieFormat.ShowInnerPoints = false;
seriesB.SerieFormat.ShowOutlierPoints = true;
seriesB.SerieFormat.ShowMeanMarkers = true;
seriesB.SerieFormat.ShowMeanLine = false;
seriesB.SerieFormat.QuartileCalculationType =
ExcelQuartileCalculation.InclusiveMedian;
//Box and Whisker settings on third series
IChartSerie seriesC = chart.Series[2];
seriesC.SerieFormat.ShowInnerPoints = false;
seriesC.SerieFormat.ShowOutlierPoints = true;
seriesC.SerieFormat.ShowMeanMarkers = true;
seriesC.SerieFormat.ShowMeanLine = false;
seriesC.SerieFormat.QuartileCalculationType =
ExcelQuartileCalculation.ExclusiveMedian;
workbook.SaveAs("Box and Whisker.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Create a chart
Dim chart As IChartShape = sheet.Charts.Add()
'Set the chart title
chart.ChartTitle = "Test Scores"
'Set chart type as Box and Whisker
chart.ChartType = ExcelChartType.BoxAndWhisker
'Set data range in the worksheet
chart.DataRange = sheet("A1:D16")
'Box and Whisker settings on first series
Dim seriesA As IChartSerie = chart.Series(0)
seriesA.SerieFormat.ShowInnerPoints = False
seriesA.SerieFormat.ShowOutlierPoints = True
seriesA.SerieFormat.ShowMeanMarkers = True
seriesA.SerieFormat.ShowMeanLine = False
seriesA.SerieFormat.QuartileCalculationType =
ExcelQuartileCalculation.ExclusiveMedian
'Box and Whisker settings on second series
Dim seriesB As IChartSerie = chart.Series(1)
seriesB.SerieFormat.ShowInnerPoints = False
seriesB.SerieFormat.ShowOutlierPoints = True
seriesB.SerieFormat.ShowMeanMarkers = True
seriesB.SerieFormat.ShowMeanLine = False
seriesB.SerieFormat.QuartileCalculationType =
ExcelQuartileCalculation.InclusiveMedian
'Box and Whisker settings on third series
Dim seriesC As IChartSerie = chart.Series(2)
seriesC.SerieFormat.ShowInnerPoints = False
seriesC.SerieFormat.ShowOutlierPoints = True
seriesC.SerieFormat.ShowMeanMarkers = True
seriesC.SerieFormat.ShowMeanLine = False

```

```

seriesC.SerieFormat.QuartileCalculationType =
ExcelQuartileCalculation.ExclusiveMedian
workbook.SaveAs ("Box and Whisker.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set the chart title
    chart.ChartTitle = "Test Scores";
    //Set chart type as Box and Whisker
    chart.ChartType = ExcelChartType.BoxAndWhisker;
    //Set data range in the worksheet
    chart.DataRange = sheet["A1:D16"];
    //Box and Whisker settings on first series
    IChartSerie seriesA = chart.Series[0];
    seriesA.SerieFormat.ShowInnerPoints = false;
    seriesA.SerieFormat.ShowOutlierPoints = true;
    seriesA.SerieFormat.ShowMeanMarkers = true;
    seriesA.SerieFormat.ShowMeanLine = false;
    seriesA.SerieFormat.QuartileCalculationType =
    ExcelQuartileCalculation.ExclusiveMedian;
    //Box and Whisker settings on second series
    IChartSerie seriesB = chart.Series[1];
    seriesB.SerieFormat.ShowInnerPoints = false;
    seriesB.SerieFormat.ShowOutlierPoints = true;
    seriesB.SerieFormat.ShowMeanMarkers = true;
    seriesB.SerieFormat.ShowMeanLine = false;
    seriesB.SerieFormat.QuartileCalculationType =
    ExcelQuartileCalculation.InclusiveMedian;
    //Box and Whisker settings on third series
    IChartSerie seriesC = chart.Series[2];
    seriesC.SerieFormat.ShowInnerPoints = false;
    seriesC.SerieFormat.ShowOutlierPoints = true;
    seriesC.SerieFormat.ShowMeanMarkers = true;
    seriesC.SerieFormat.ShowMeanLine = false;
    seriesC.SerieFormat.QuartileCalculationType =
    ExcelQuartileCalculation.ExclusiveMedian;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;

```

```

savePicker.SuggestedFileName = "Box and Whisker";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set the chart title
    chart.ChartTitle = "Test Scores";
    //Set chart type as Box and Whisker
    chart.ChartType = ExcelChartType.BoxAndWhisker;
    //Set data range in the worksheet
    chart.DataRange = sheet["A1:D16"];
    //Box and Whisker settings on first series
    IChartSerie seriesA = chart.Series[0];
    seriesA.SerieFormat.ShowInnerPoints = false;
    seriesA.SerieFormat.ShowOutlierPoints = true;
    seriesA.SerieFormat.ShowMeanMarkers = true;
    seriesA.SerieFormat.ShowMeanLine = false;
    seriesA.SerieFormat.QuartileCalculationType =
    ExcelQuartileCalculation.ExclusiveMedian;
    //Box and Whisker settings on second series
    IChartSerie seriesB = chart.Series[1];
    seriesB.SerieFormat.ShowInnerPoints = false;
    seriesB.SerieFormat.ShowOutlierPoints = true;
    seriesB.SerieFormat.ShowMeanMarkers = true;
    seriesB.SerieFormat.ShowMeanLine = false;
    seriesB.SerieFormat.QuartileCalculationType =
    ExcelQuartileCalculation.InclusiveMedian;
    //Box and Whisker settings on third series
    IChartSerie seriesC = chart.Series[2];
    seriesC.SerieFormat.ShowInnerPoints = false;
    seriesC.SerieFormat.ShowOutlierPoints = true;
    seriesC.SerieFormat.ShowMeanMarkers = true;
    seriesC.SerieFormat.ShowMeanLine = false;
    seriesC.SerieFormat.QuartileCalculationType =
    ExcelQuartileCalculation.ExclusiveMedian;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Box and Whisker.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
}

```

```
stream.Dispose();
}
```

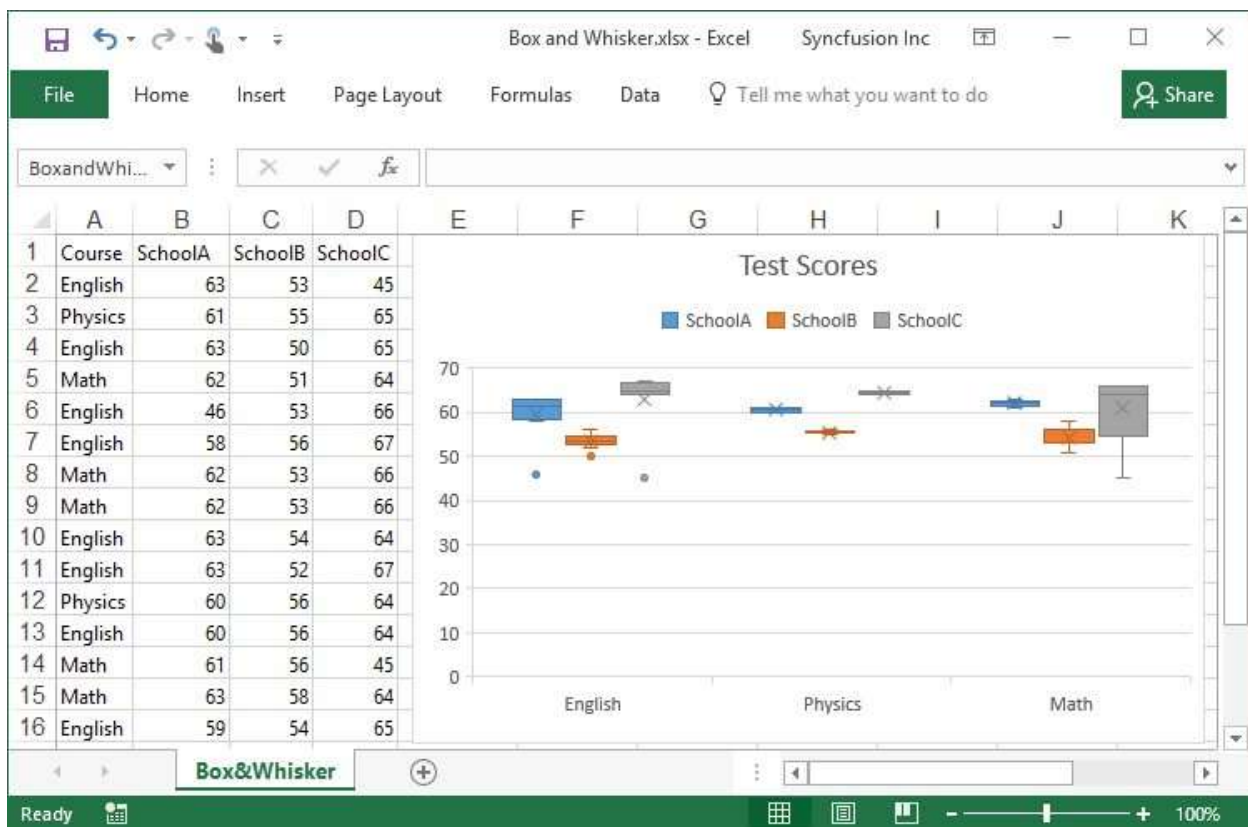
XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelParseOptions.ParseWorksheetsOnDemand);
    IWorksheet sheet = workbook.Worksheets[0];
    ///Create a chart
    IChartShape chart = sheet.Charts.Add();
    ///Set the chart title
    chart.ChartTitle = "Test Scores";
    ///Set chart type as Box and Whisker
    chart.ChartType = ExcelChartType.BoxAndWhisker;
    ///Set data range in the worksheet
    chart.DataRange = sheet["A1:D16"];
    ///Box and Whisker settings on first series
    IChartSerie seriesA = chart.Series[0];
    seriesA.SerieFormat.ShowInnerPoints = false;
    seriesA.SerieFormat.ShowOutlierPoints = true;
    seriesA.SerieFormat.ShowMeanMarkers = true;
    seriesA.SerieFormat.ShowMeanLine = false;
    seriesA.SerieFormat.QuartileCalculationType =
    ExcelQuartileCalculation.ExclusiveMedian;
    ///Box and Whisker settings on second series
    IChartSerie seriesB = chart.Series[1];
    seriesB.SerieFormat.ShowInnerPoints = false;
    seriesB.SerieFormat.ShowOutlierPoints = true;
    seriesB.SerieFormat.ShowMeanMarkers = true;
    seriesB.SerieFormat.ShowMeanLine = false;
    seriesB.SerieFormat.QuartileCalculationType =
    ExcelQuartileCalculation.InclusiveMedian;
    ///Box and Whisker settings on third series
    IChartSerie seriesC = chart.Series[2];
    seriesC.SerieFormat.ShowInnerPoints = false;
    seriesC.SerieFormat.ShowOutlierPoints = true;
    seriesC.SerieFormat.ShowMeanMarkers = true;
    seriesC.SerieFormat.ShowMeanLine = false;
    seriesC.SerieFormat.QuartileCalculationType =
    ExcelQuartileCalculation.ExclusiveMedian;
    ///Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    ///Save the document as file and view the saved document
```

```
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Box
and Whisker.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Box and
Whisker.xlsx", "application/msexcel", stream);
}
}
```

The input template can be downloaded [here](#).

The following screen shot shows the output of above code.



Waterfall

Waterfall chart helps to quickly understand the finances of business owners by viewing profit and loss statements. With a Waterfall chart, you can quickly illustrate the line items in your financial data and get a clear picture of how each item is impacting your bottom line.

Following code example illustrates how to create Waterfall chart.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as Waterfall
    chart.ChartType = ExcelChartType.WaterFall;
    //Set data range in the worksheet
    chart.DataRange = sheet["A2:B8"];
    //Data point settings as total in chart
    chart.Series[0].DataPoints[3].SetAsTotal = true;
    chart.Series[0].DataPoints[6].SetAsTotal = true;
    //Showing the connector lines between data points
    chart.Series[0].SerieFormat.ShowConnectorLines = true;
    //Set the chart title
    chart.ChartTitle = "Company Profit (in USD)";
    //Formatting data label and legend option
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
    chart.Legend.Position = ExcelLegendPosition.Right;
    workbook.SaveAs("Waterfall.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
    ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Create a chart
Dim chart As IChartShape = sheet.Charts.Add()
'Set chart type as Waterfall
chart.ChartType = ExcelChartType.WaterFall
'Set data range in the worksheet
chart.DataRange = sheet("A2:B8")
'Datapoint settings as total in chart
chart.Series(0).DataPoints(3).SetAsTotal = True
chart.Series(0).DataPoints(6).SetAsTotal = True
'Showing the connector lines between data points
chart.Series(0).SerieFormat.ShowConnectorLines = True
'Set the chart title
chart.ChartTitle = "Company Profit (in USD)"
'Formatting data label and legend option
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.IsValue = True
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.Size = 8
chart.Legend.Position = ExcelLegendPosition.Right
workbook.SaveAs("Waterfall.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as Waterfall
    chart.ChartType = ExcelChartType.WaterFall;
    //Set data range in the worksheet
    chart.DataRange = sheet["A2:B8"];
    //Data point settings as total in chart
    chart.Series[0].DataPoints[3].SetAsTotal = true;
    chart.Series[0].DataPoints[6].SetAsTotal = true;
    //Showing the connector lines between data points
    chart.Series[0].SerieFormat.ShowConnectorLines = true;
    //Set the chart title
    chart.ChartTitle = "Company Profit (in USD)";
    //Formatting data label and legend option
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
    chart.Legend.Position = ExcelLegendPosition.Right;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Waterfall";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook =
    application.Workbooks.Open(inputStream, ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
}

```

```
//Create a chart
IChartShape chart = sheet.Charts.Add();
//Set chart type as Waterfall
chart.ChartType = ExcelChartType.WaterFall;
//Set data range in the worksheet
chart.DataRange = sheet["A2:B8"];
//Data point settings as total in chart
chart.Series[0].DataPoints[3].SetAsTotal = true;
chart.Series[0].DataPoints[6].SetAsTotal = true;
//Showing the connector lines between data points
chart.Series[0].SerieFormat.ShowConnectorLines = true;
//Set the chart title
chart.ChartTitle = "Company Profit (in USD)";
//Formatting data label and legend option
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
chart.Legend.Position = ExcelLegendPosition.Right;
//Saving the workbook as stream
FileStream stream = new FileStream("Waterfall.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook =
    application.Workbooks.Open(inputStream, ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as Waterfall
    chart.ChartType = ExcelChartType.WaterFall;
    //Set data range in the worksheet
    chart.DataRange = sheet["A2:B8"];
    //Data point settings as total in chart
    chart.Series[0].DataPoints[3].SetAsTotal = true;
    chart.Series[0].DataPoints[6].SetAsTotal = true;
    //Showing the connector lines between data points
    chart.Series[0].SerieFormat.ShowConnectorLines = true;
    //Set the chart title
    chart.ChartTitle = "Company Profit (in USD)";
    //Formatting data label and legend option
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.IsValue = true;
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
    chart.Legend.Position = ExcelLegendPosition.Right;
    //Saving the workbook as stream
}
```



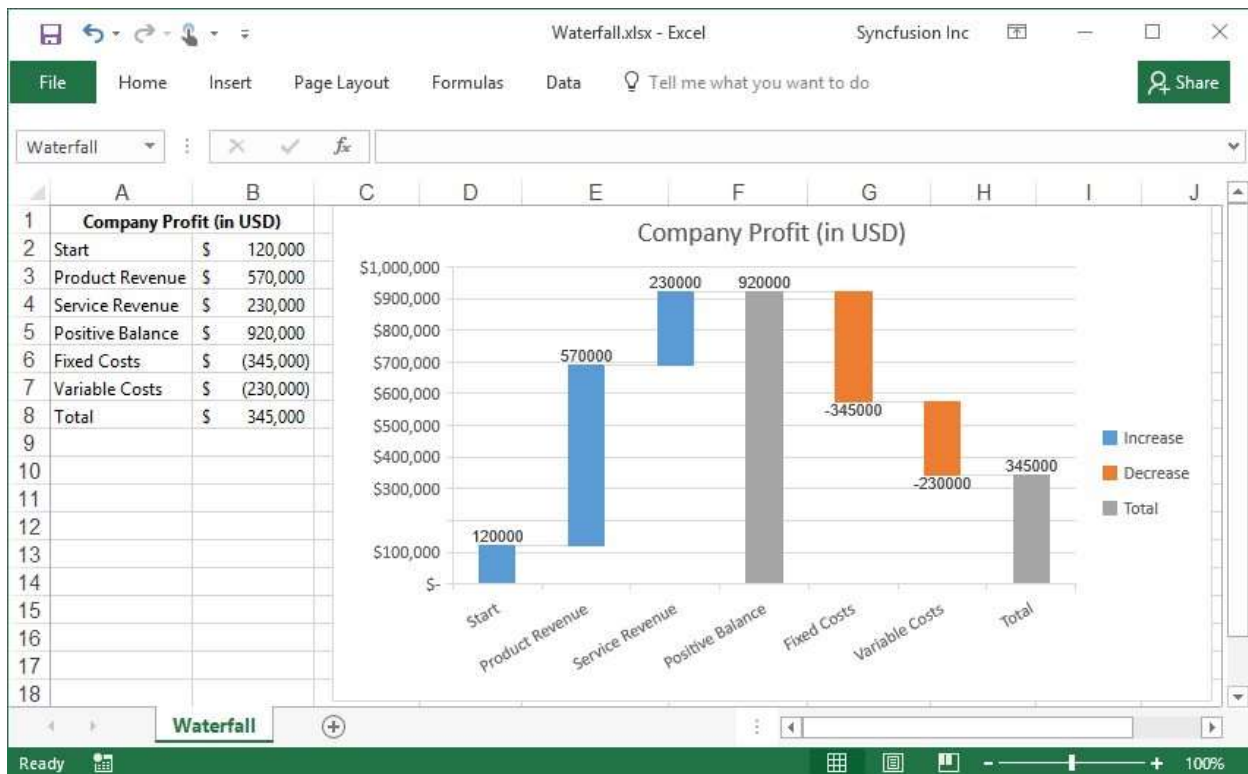
```

MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Waterfall
all.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Waterfall.xlsx",
"application/msexcel", stream);
}
}

```

The input template can be downloaded [here](#).

The following screen shot shows the output of above code.



Histogram

[Histogram](#) shows the frequencies within a distribution. Each column of the chart is called a bin, which can be changed further to analyze the data.

Following code example illustrates how to create Histogram.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as Histogram
    chart.ChartType = ExcelChartType.Histogram;
    //Set data range in the worksheet
    chart.DataRange = sheet["A1:A15"];
    //Category axis bin settings
    chart.PrimaryCategoryAxis.BinWidth = 8;
    //Gap width settings
    chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
    //Set the chart title and axis title
    chart.ChartTitle = "Height Data";
    chart.PrimaryValueAxis.Title = "Number of students";
    chart.PrimaryCategoryAxis.Title = "Height";
    //Hiding the legend
    chart.HasLegend = false;
    workbook.SaveAs("Histogram.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
    ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Create a chart
Dim chart As IChartShape = sheet.Charts.Add()
'Set chart type as Histogram
chart.ChartType = ExcelChartType.Histogram
'Set data range in the worksheet
chart.DataRange = sheet("A1:A15")
'Category axis bin settings
chart.PrimaryCategoryAxis.BinWidth = 8
'Gap width settings
chart.Series(0).SerieFormat.CommonSerieOptions.GapWidth = 6
'Set the chart title and axis title
chart.ChartTitle = "Height Data"
chart.PrimaryValueAxis.Title = "Number of students"
chart.PrimaryCategoryAxis.Title = "Height"
'Hiding the legend
chart.HasLegend = False
workbook.SaveAs("Histogram.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as Histogram
    chart.ChartType = ExcelChartType.Histogram;
    //Set data range in the worksheet
    chart.DataRange = sheet["A1:A15"];
    //Category axis bin settings
    chart.PrimaryCategoryAxis.BinWidth = 8;
    //Gap width settings
    chart.Series[0].SeriesFormat.CommonSerieOptions.GapWidth = 6;
    //Set the chart title and axis title
    chart.ChartTitle = "Height Data";
    chart.PrimaryValueAxis.Title = "Number of students";
    chart.PrimaryCategoryAxis.Title = "Height";
    //Hiding the legend
    chart.HasLegend = false;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Histogram";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart

```

```

IChartShape chart = sheet.Charts.Add();
//Set chart type as Histogram
chart.ChartType = ExcelChartType.Histogram;
//Set data range in the worksheet
chart.DataRange = sheet["A1:A15"];
//Category axis bin settings
chart.PrimaryCategoryAxis.BinWidth = 8;
//Gap width settings
chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
//Set the chart title and axis title
chart.ChartTitle = "Height Data";
chart.PrimaryValueAxis.Title = "Number of students";
chart.PrimaryCategoryAxis.Title = "Height";
//Hiding the legend
chart.HasLegend = false;
//Saving the workbook as stream
FileStream stream = new FileStream("Histogram.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

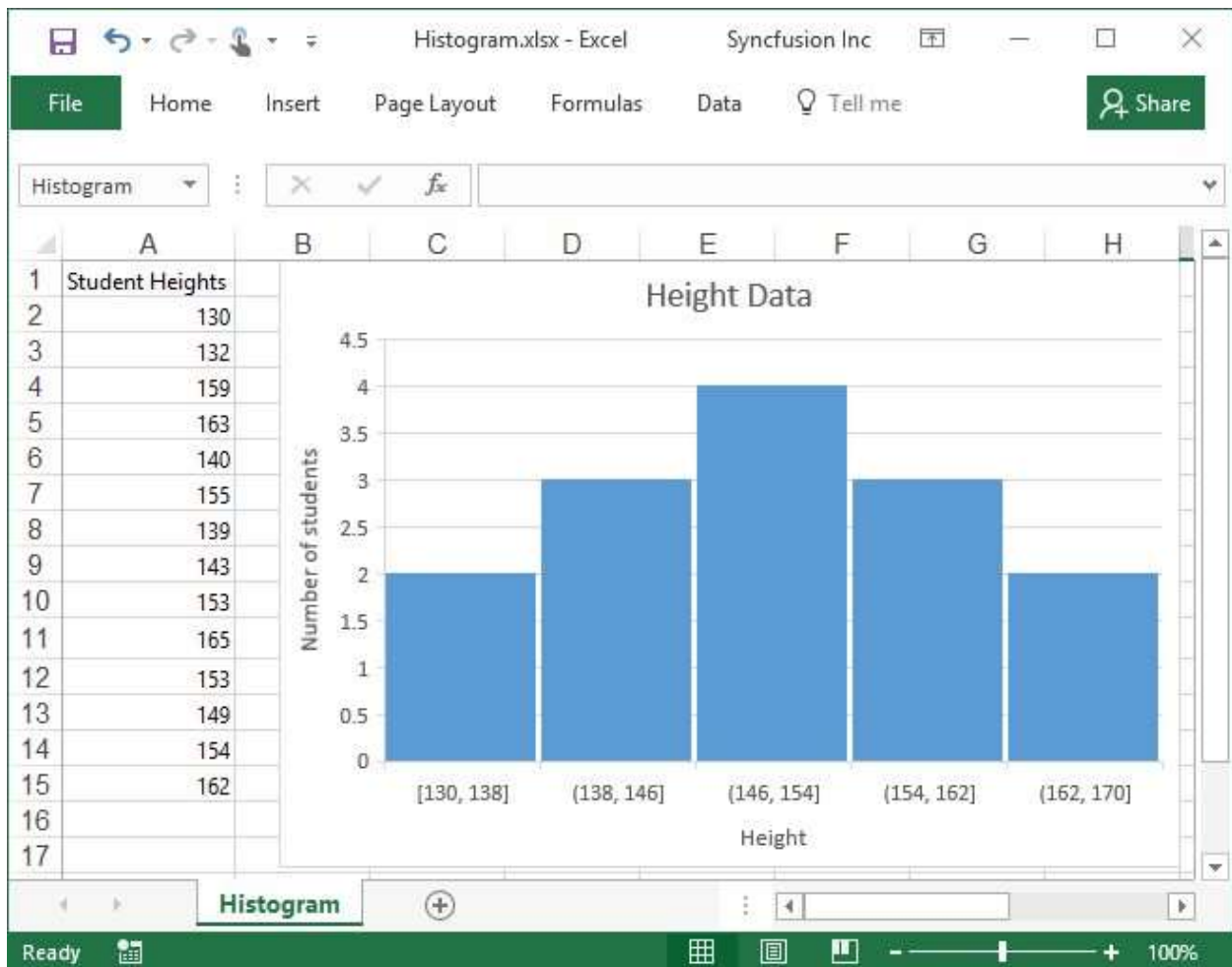
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as Histogram
    chart.ChartType = ExcelChartType.Histogram;
    //Set data range in the worksheet
    chart.DataRange = sheet["A1:A15"];
    //Category axis bin settings
    chart.PrimaryCategoryAxis.BinWidth = 8;
    //Gap width settings
    chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
    //Set the chart title and axis title
    chart.ChartTitle = "Height Data";
    chart.PrimaryValueAxis.Title = "Number of students";
    chart.PrimaryCategoryAxis.Title = "Height";
    //Hiding the legend
    chart.HasLegend = false;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
}

```

```
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Histogram.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Histogram.xlsx", "application/msexcel", stream);
}
}
```

The input template can be downloaded [here](#).

The following screen shot shows the output of above code.



Pareto

Pareto is a sorted histogram where columns sorted in descending order and a line representing the cumulative total percentage.

Following code example illustrates how to create Pareto chart.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as Pareto
    chart.ChartType = ExcelChartType.Pareto;
    //Set data range in the worksheet
    chart.DataRange = sheet["A2:B8"];
    //Set category values as bin values
    chart.PrimaryCategoryAxis.IsBinningByCategory = true;
    //Formatting Pareto line
    chart.Series[0].ParetoLineFormat.LineProperties.ColorIndex =
        ExcelKnownColors.Bright_green;
    //Gap width settings
    chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
    //Set the chart title
    chart.ChartTitle = "Expenses";
    //Hiding the legend
    chart.HasLegend = false;
    workbook.SaveAs("Pareto.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
    ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Create a chart
Dim chart As IChartShape = sheet.Charts.Add()
'Set chart type as Pareto
chart.ChartType = ExcelChartType.Pareto
'Set data range in the worksheet
chart.DataRange = sheet("A2:B8")
'Set category axis as bin option
chart.PrimaryCategoryAxis.IsBinningByCategory = True
'Formatting Pareto line
chart.Series(0).ParetoLineFormat.LineProperties.ColorIndex =
    ExcelKnownColors.Bright_green
'Gap width settings
chart.Series(0).SerieFormat.CommonSerieOptions.GapWidth = 6
'Set the chart title
```

```

chart.ChartTitle = "Expenses"
'Hiding the legend
chart.HasLegend = False
workbook.SaveAs("Pareto.xlsx")
End Using

```

UWP

```

ExcelEngine excelEngine = new ExcelEngine();
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as Pareto
    chart.ChartType = ExcelChartType.Pareto;
    //Set data range in the worksheet
    chart.DataRange = sheet["A2:B8"];
    //Set category values as bin values
    chart.PrimaryCategoryAxis.IsBinningByCategory = true;
    //Formatting Pareto line
    chart.Series[0].ParetoLineFormat.LineProperties.ColorIndex =
        ExcelKnownColors.Bright_green;
    //Gap width settings
    chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
    //Set the chart title
    chart.ChartTitle = "Expenses";
    //Hiding the legend
    chart.HasLegend = false;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Pareto";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{

```

```

IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
//Create a chart
IChartShape chart = sheet.Charts.Add();
//Set chart type as Pareto
chart.ChartType = ExcelChartType.Pareto;
//Set data range in the worksheet
chart.DataRange = sheet["A2:B8"];
//Set category values as bin values
chart.PrimaryCategoryAxis.IsBinningByCategory = true;
//Formatting Pareto line
chart.Series[0].ParetoLineFormat.LineProperties.ColorIndex =
ExcelKnownColors.Bright_green;
//Gap width settings
chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
//Set the chart title
chart.ChartTitle = "Expenses";
//Hiding the legend
chart.HasLegend = false;
//Saving the workbook as stream
FileStream stream = new FileStream("Pareto.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
ple.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
//Create a chart
IChartShape chart = sheet.Charts.Add();
//Set chart type as Pareto
chart.ChartType = ExcelChartType.Pareto;
//Set data range in the worksheet
chart.DataRange = sheet["A2:B8"];
//Set category values as bin values
chart.PrimaryCategoryAxis.IsBinningByCategory = true;
//Formatting Pareto line
chart.Series[0].ParetoLineFormat.LineProperties.ColorIndex =
ExcelKnownColors.Bright_green;
}

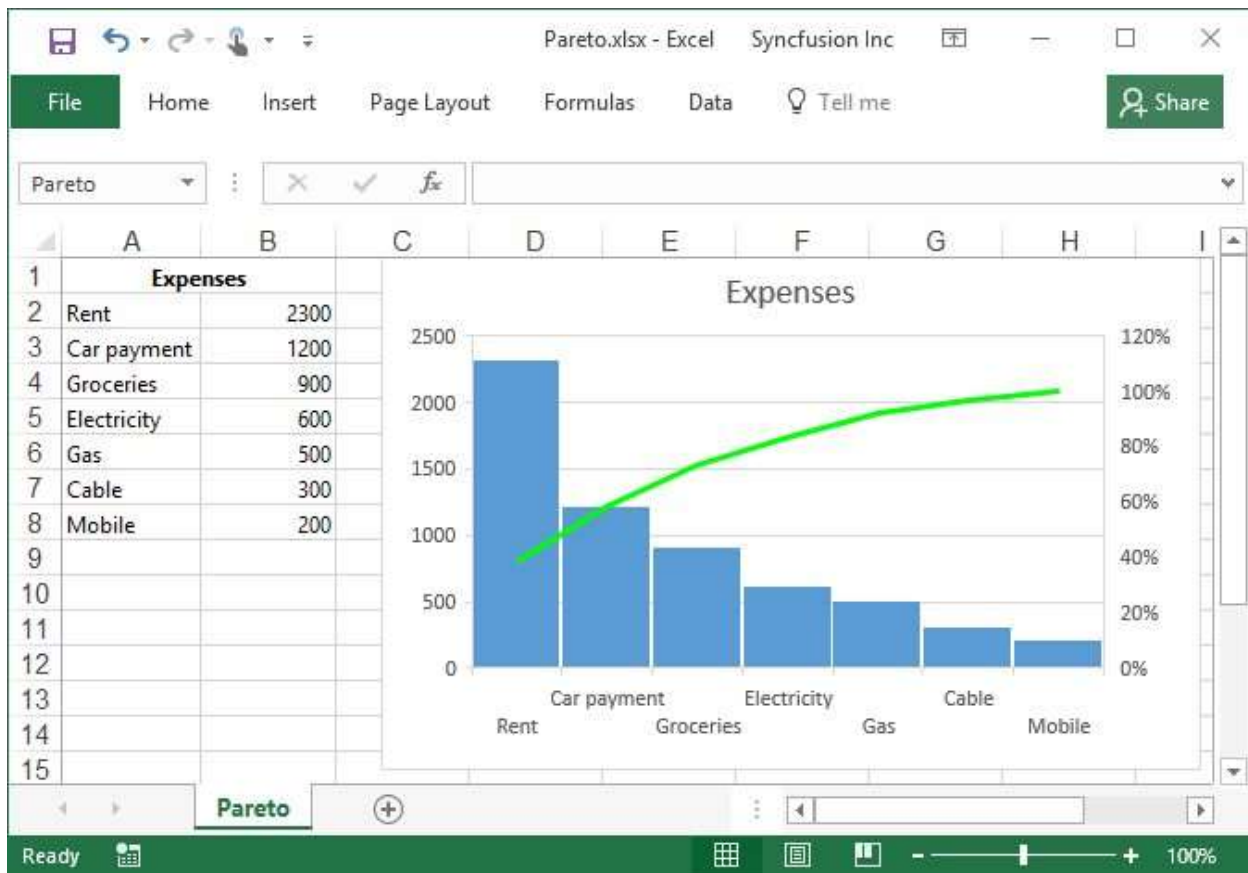
```



```
//Gap width settings
chart.Series[0].SerieFormat.CommonSerieOptions.GapWidth = 6;
//Set the chart title
chart.ChartTitle = "Expenses";
//Hiding the legend
chart.HasLegend = false;
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Pareto
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Pareto.xlsx",
"application/msexcel", stream);
}
}
```

The input template can be downloaded [here](#).

The following screen shot shows the output of above code.



TreeMap

[TreeMap](#) provides a hierarchical view of data as clustered rectangle with a specific weighted attribute determining the size of the rectangle.

Following code example illustrates how to create TreeMap chart.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as TreeMap
    chart.ChartType = ExcelChartType.TreeMap;
    //Set data range in the worksheet
    chart.DataRange = sheet["A2:C11"];
    //Set the chart title
    chart.ChartTitle = "Area by countries";
    //Set the TreeMap label option
    chart.Series[0].SeriesFormat.TreeMapLabelOption =
        ExcelTreeMapLabelOption.Banner;
    //Formatting data labels
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
}
```

```
workbook.SaveAs ("Treemap.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Create a chart
Dim chart As IChartShape = sheet.Charts.Add()
'Set chart type as TreeMap
chart.ChartType = ExcelChartType.TreeMap
'Set data range in the worksheet
chart.DataRange = sheet("A2:C11")
'Set the chart title
chart.ChartTitle = "Area by countries"
'Set the Treemap label option
chart.Series(0).SerieFormat.TreeMapLabelOption =
ExcelTreeMapLabelOption.Banner
'Formatting data labels
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.Size = 8
workbook.SaveAs ("Treemap.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
//Create a chart
IChartShape chart = sheet.Charts.Add();
//Set chart type as TreeMap
chart.ChartType = ExcelChartType.TreeMap;
//Set data range in the worksheet
chart.DataRange = sheet["A2:C11"];
//Set the chart title
chart.ChartTitle = "Area by countries";
//Set the Treemap label option
chart.Series[0].SerieFormat.TreeMapLabelOption =
ExcelTreeMapLabelOption.Banner;
//Formatting data labels
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
```

```
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Treemap";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as TreeMap
    chart.ChartType = ExcelChartType.TreeMap;
    //Set data range in the worksheet
    chart.DataRange = sheet["A2:C11"];
    //Set the chart title
    chart.ChartTitle = "Area by countries";
    //Set the Treemap label option
    chart.Series[0].SerieFormat.TreeMapLabelOption =
    ExcelTreeMapLabelOption.Banner;
    //Formatting data labels
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Treemap.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

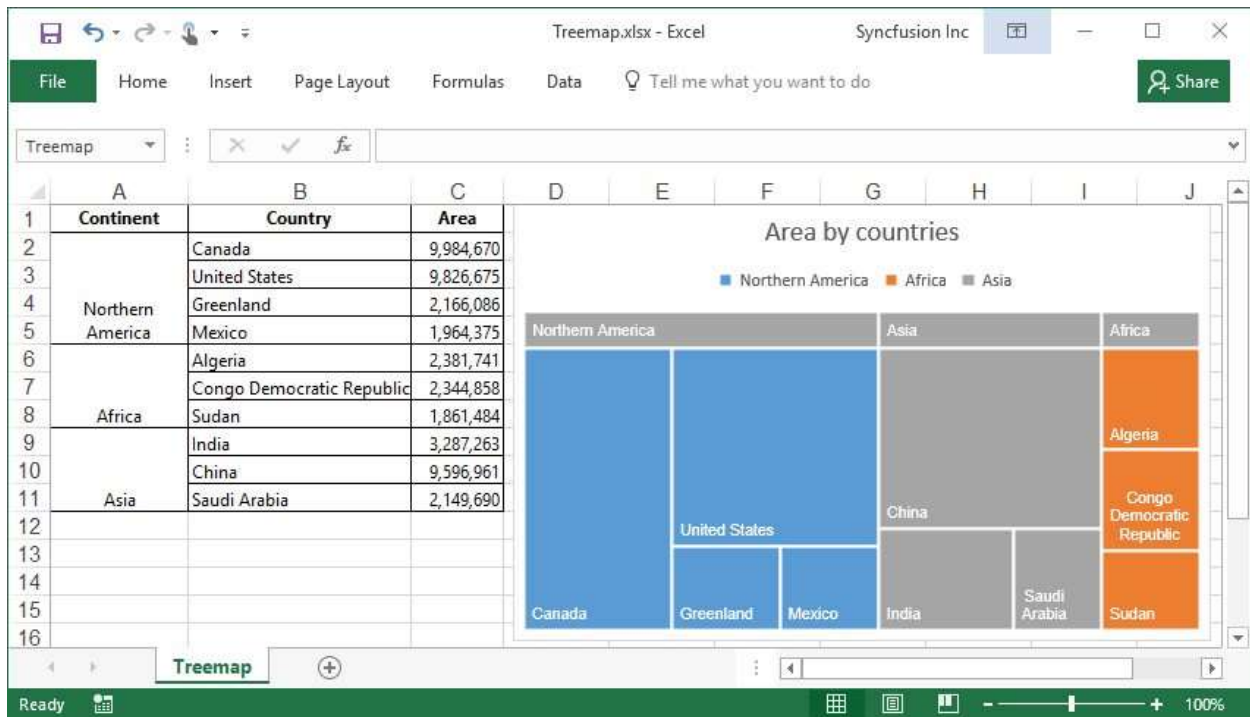
XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sam
    ple.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
}
```

```
IWorksheet sheet = workbook.Worksheets[0];
//Create a chart
IChartShape chart = sheet.Charts.Add();
//Set chart type as TreeMap
chart.ChartType = ExcelChartType.TreeMap;
//Set data range in the worksheet
chart.DataRange = sheet["A2:C11"];
//Set the chart title
chart.ChartTitle = "Area by countries";
//Set the Treemap label option
chart.Series[0].SerieFormat.TreeMapLabelOption =
ExcelTreeMapLabelOption.Banner;
//Formatting data labels
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Treema
p.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Treemap.xlsx",
"application/msexcel", stream);
}
}
```

The input template can be downloaded [here](#).

The following screen shot shows the output of above code.



Sunburst

Sunburst provides a hierarchical view of data where each level of the hierarchy is represented by one ring or circle with the innermost circle as the top of the hierarchy.

Following code example illustrates how to create Sunburst chart.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as Sunburst
    chart.ChartType = ExcelChartType.SunBurst;
    //Set data range in the worksheet
    chart.DataRange = sheet["A1:D16"];
    //Set the chart title
    chart.ChartTitle = "Sales by annual";
    //Formatting data labels
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
    //Hiding the legend
    chart.HasLegend = false;
    workbook.SaveAs("Sunburst.xlsx");
}
```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Create a chart
Dim chart As IChartShape = sheet.Charts.Add()
'Set chart type as Sunburst
chart.ChartType = ExcelChartType.SunBurst
'Set data range in the worksheet
chart.DataRange = sheet("A1:D16")
'Set the chart title
chart.ChartTitle = "Sales by annual"
'Formatting data labels
chart.Series(0).DataPoints.DefaultDataPoint.DataLabels.Size = 8
'Hiding the legend
chart.HasLegend = False
workbook.SaveAs("Sunburst.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
//Create a chart
IChartShape chart = sheet.Charts.Add();
//Set chart type as Sunburst
chart.ChartType = ExcelChartType.SunBurst;
//Set data range in the worksheet
chart.DataRange = sheet["A1:D16"];
//Set the chart title
chart.ChartTitle = "Sales by annual";
//Formatting data labels
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
//Hiding the legend
chart.HasLegend = false;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Sunburst";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker

```

```
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as Sunburst
    chart.ChartType = ExcelChartType.SunBurst;
    //Set data range in the worksheet
    chart.DataRange = sheet["A1:D16"];
    //Set the chart title
    chart.ChartTitle = "Sales by annual";
    //Formatting data labels
    chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;
    //Hiding the legend
    chart.HasLegend = false;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Sunburst.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

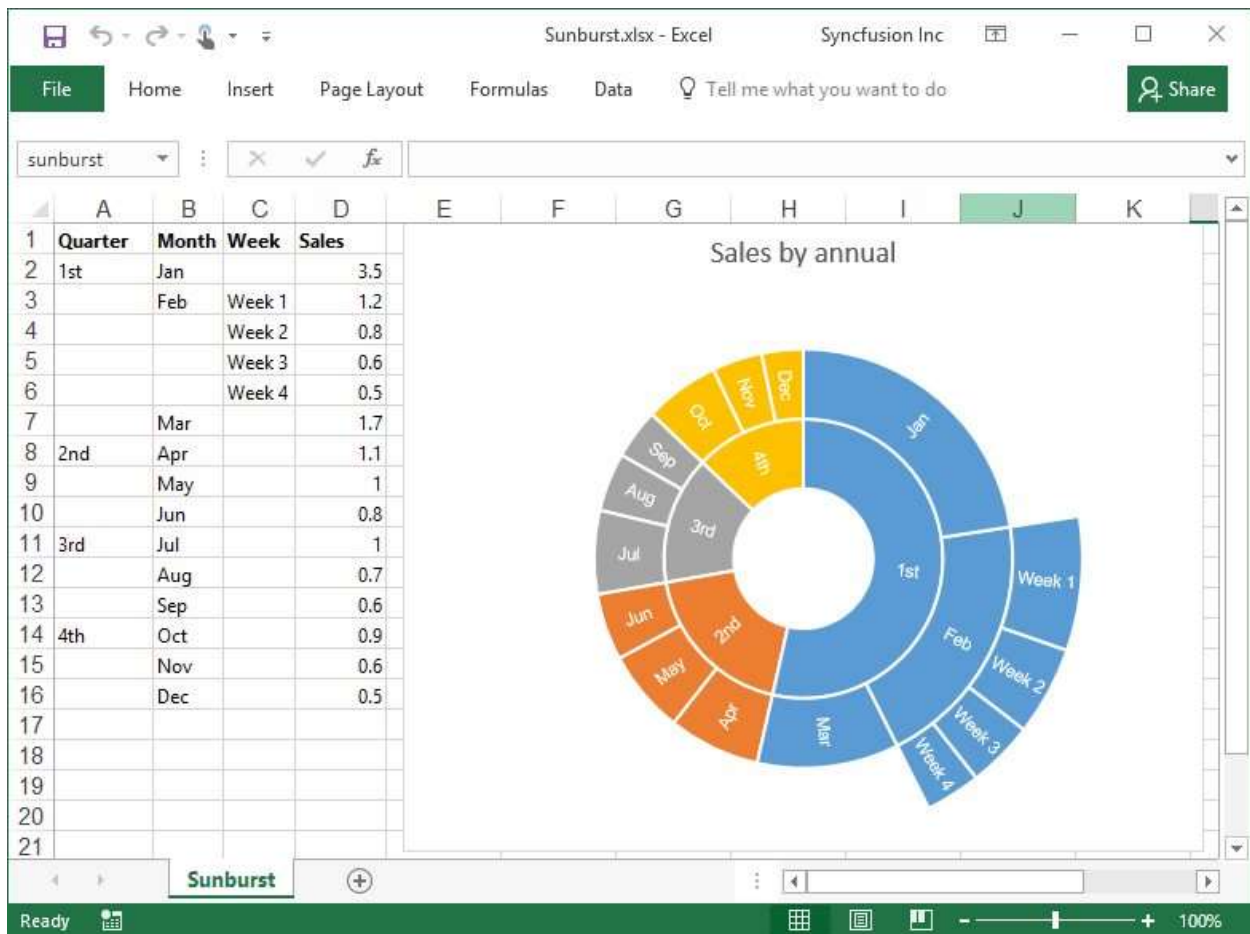
```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    // "App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a chart
    IChartShape chart = sheet.Charts.Add();
    //Set chart type as Sunburst
    chart.ChartType = ExcelChartType.SunBurst;
    //Set data range in the worksheet
    chart.DataRange = sheet["A1:D16"];
    //Set the chart title
}
```



```
chart.ChartTitle = "Sales by annual";  
//Formatting data labels  
chart.Series[0].DataPoints.DefaultDataPoint.DataLabels.Size = 8;  
//Hiding the legend  
chart.HasLegend = false;  
//Saving the workbook as stream  
MemoryStream stream = new MemoryStream();  
workbook.SaveAs(stream);  
stream.Position = 0;  
//Save the document as file and view the saved document  
//The operation in SaveAndView under Xamarin varies between Windows Phone,  
//Android and iOS platforms. Please refer xlsio/xamarin section for respective  
//code samples.  
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==  
TargetPlatform.Windows)  
{  
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Sunburst.xlsx", "application/msexcel", stream);  
}  
else  
{  
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Sunburst.xlsx",  
"application/msexcel", stream);  
}  
}
```

The input template can be downloaded [here](#).

The following screen shot shows the output of above code.



Note: These Charts are supported only in Excel 2016 and are not visible in the earlier versions.

Supported Chart Types

The following chart types are supported in XlsIO.

- [Area](#)
- [Area 3D](#)
- [Area Stacked](#)
- [AreaStacked100](#)
- [AreaStacked100 3D](#)
- [AreaStacked3D](#)
- [Bar Clustered](#)
- [BarClustered3D](#)
- [Bar Stacked](#)
- [BarStacked100](#)
- [BarStacked100 3D](#)
- [BarStacked3D](#)
- [Bubble](#)
- [Bubble 3D](#)
- [Column 3D](#)
- [Column Clustered](#)
- [ColumnClustered3D](#)

- [Column Stacked](#)
- [ColumnStacked100](#)
- [ColumnStacked100 3D](#)
- [ColumnStacked3D](#)
- [Combination Chart](#)
- [ConeBarClustered](#)
- [ConeBarStacked](#)
- [ConeBarStacked 100](#)
- [Cone Clustered](#)
- [ConeClustered3D](#)
- [Cone Stacked](#)
- [ConeStacked100](#)
- [CylinderBarClustered](#)
- [CylinderBarStacked](#)
- [CylinderBarStacked 100](#)
- [Cylinder Clustered](#)
- [CylinderClustered3D](#)
- [Cylinder Stacked](#)
- [CylinderStacked100](#)
- [Doughnut](#)
- [Doughnut Exploded](#)
- [Line](#)
- [Line 3D](#)
- [Line Markers](#)
- [LineMarkersStacked](#)
- [LineMarkersStacked 100](#)
- [Line Stacked](#)
- [LineStacked100](#)
- [Pie](#)
- [Pie 3D](#)
- [Pie Bar](#)
- [Pie Exploded](#)
- [PieExploded3D](#)
- [PieOfPie](#)
- [PyramidBarClustered](#)
- [PyramidBarStacked](#)
- [PyramidBarStacked 100](#)
- [Pyramid Clustered](#)
- [PyramidClustered3D](#)
- [Pyramid Stacked](#)
- [PyramidStacked100](#)
- [Radar](#)
- [Radar Filled](#)
- [Radar Markers](#)
- [Scatter Line](#)
- [ScatterLineMarkers](#)
- [Scatter Markers](#)
- [Scatter SmoothedLine](#)

- [ScatterSmoothedLineMarkers](#)
- [Stock_HighLowClose](#)
- [Stock_OpenHighLowClose](#)
- [Stock_VolumeHighLowClose](#)
- [Stock_VolumeOpenHighLowClose](#)
- [Surface_3D](#)
- [Surface_Contour](#)
- [SurfaceNoColor3D](#)
- [SurfaceNoColorContour](#)
- [Funnel](#)
- [Box and Whisker](#)
- [Waterfall](#)
- [Histogram](#)
- [Pareto](#)
- [Treemap](#)
- [Sunburst](#)

Working with Template Markers

A template marker is a special marker symbol created in an Excel template that appends multiple records from a data source into a worksheet. This marker automatically maps the column name in the data source and names of the marker fields in the template Excel document and fills the data (text or image). Essential XlsIO allows you to bind the template markers to data from various sources, such as

- DataTable
- Collection objects
- Nested collection objects
- Arrays

Template marker Syntax

Each marker starts with a prefix character (by default it is “%” character). The marker is followed by the variable name and properties which are delimited by a character (by default it is semicolon “;”).

```
%<MarkerVariable>.<Property>
```

For example: %Customers.CompanyName

Where, “Customers” is marker variable name and CompanyName is the property name

You can change the marker prefix and delimiter characters by the **MarkerPrefix** and **ArgumentSeparator** properties of the ITemplateMarkersProcessor instance respectively.

Arguments

You can specify the following arguments with the marker to customize the worksheet.

Horizontal-This argument specifies the horizontal direction of the data filling for variables.

Syntax: %<MarkerVariable>.<Property>;horizontal

Vertical-This argument specifies the vertical direction of the data filling for variables. By default, data will be filled in vertical direction

Syntax: %<MarkerVariable>.<Property>;vertical

insert-This argument inserts new row or column, depending on the direction argument for each new cell. Note that by default, the rows will not be inserted.

Syntax: %<MarkerVariable>.<Property>;insert

insert:copystyles-This argument copies styles from the above row or left column.

Syntax: %<MarkerVariable>.<Property>;insert:copystyles

jump:[cell reference in R1C1 notation]-This argument binds the data to the cell at the specified reference. Cell reference addresses can be relative or absolute.

Syntax: %<MarkerVariable>.<Property>;jump:R2C2

copyrange:[top-left cell reference in R1C1]:[bottom-right cell reference in R1C1]-Copies the specified cells after each cell import.

Syntax: %<MarkerVariable>.<Property>;copyrange:R2C2:R4C4

default-This argument adds the property value once per object for the corresponding records in the column while importing nested collection objects.

Syntax: %<MarkerVariable>.<Property>default

merge-This argument merges the cells in the column for each object set while importing nested collection objects.

Syntax: %<MarkerVariable>.<Property>merge

repeat-This argument repeats the property value for the corresponding records in the column while importing nested collection objects.

Syntax: %<MarkerVariable>.<Property>repeat

collapsegroup-This argument groups the cells in the column for each object set and collapse the data while importing nested collection objects.

Syntax: %<MarkerVariable>.<Property>collapsegroup

expandgroup-This argument groups the cells in the column for each object set and expand the data while importing nested collection objects.

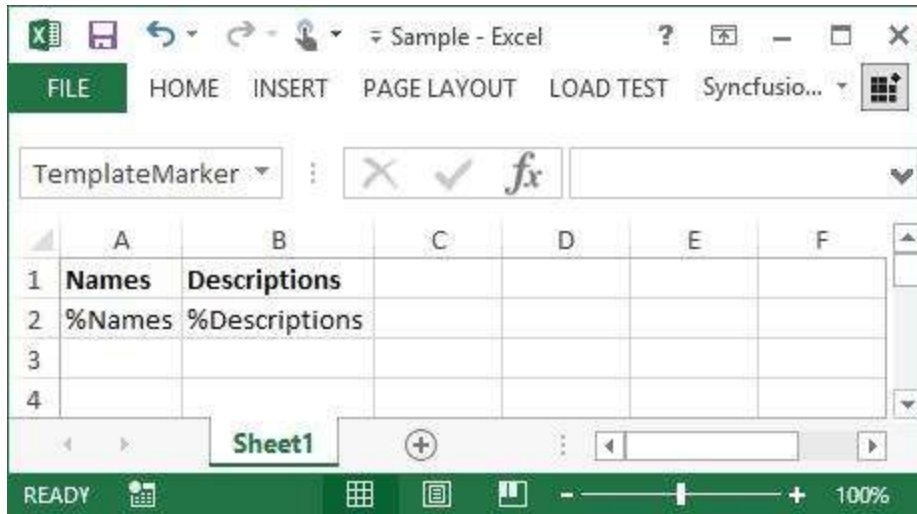
Syntax: %<MarkerVariable>.<Property>expandgroup

[Bind from Array](#)

An array of data can be binded to the marker in the template document.

Syntax: %VariableArray

The following screenshot represents the input template which has a template marker.



Following code example illustrates how to bind the data from an array to a marker.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    ExcelEngine excelEngine = new ExcelEngine();
    IWorkbook workbook = excelEngine.Excel.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
        workbook.CreateTemplateMarkersProcessor();
    //Insert Array Horizontally
    string[] names = new string[] { "Mickey", "Donald", "Tom", "Jerry" };
    string[] descriptions = new string[] { "Mouse", "Duck", "Cat", "Mouse" };
    //Add collections to the marker variables where the name should match with
    //input template
    marker.AddVariable("Names", names);
    marker.AddVariable("Descriptions", descriptions);
    //Process the markers in the template
    marker.ApplyMarkers();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs("TemplateMarker.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Open("Sample.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Create Template Marker Processor
Dim marker As ITemplateMarkersProcessor =
    workbook.CreateTemplateMarkersProcessor()
'Insert Array Horizontally
Dim names As String() = New String() {"Mickey", "Donald", "Tom", "Jerry"}
Dim descriptions As String() = New String() {"Mouse", "Duck", "Cat",
    "Mouse"}
'Add collections to the marker variables where the name should match with
input template
```

```

marker.AddVariable("Names", names)
marker.AddVariable("Descriptions", descriptions)
'Process the markers in the template
marker.ApplyMarkers()
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("TemplateMarker.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream = assembly.GetManifestResourceStream("Sample.xlsx");
    IWorkbook workbook = await
    excelEngine.Excel.Workbooks.OpenAsync(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
    workbook.CreateTemplateMarkersProcessor();
    //Insert Array Horizontally
    string[] names = new string[] { "Mickey", "Donald", "Tom", "Jerry" };
    string[] descriptions = new string[] { "Mouse", "Duck", "Cat", "Mouse" };
    //Add collections to the marker variables where the name should match with
    input template
    marker.AddVariable("Names", names);
    marker.AddVariable("Descriptions", descriptions);
    //Process the markers in the template
    marker.ApplyMarkers();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "TemplateMarker";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
    workbook.CreateTemplateMarkersProcessor();
    //Insert Array Horizontally

```

```

string[] names = new string[] { "Mickey", "Donald", "Tom", "Jerry" };
string[] descriptions = new string[] { "Mouse", "Duck", "Cat", "Mouse" };
//Add collections to the marker variables where the name should match with
input template
marker.AddVariable("Names", names);
marker.AddVariable("Descriptions", descriptions);
//Process the markers in the template
marker.ApplyMarkers();
//Saving the workbook as stream
workbook.Version = ExcelVersion.Excel2013;
FileStream stream = new FileStream("TemplateMarker.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream = assembly.GetManifestResourceStream("Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
    workbook.CreateTemplateMarkersProcessor();
    //Insert Array Horizontally
    string[] names = new string[] { "Mickey", "Donald", "Tom", "Jerry" };
    string[] descriptions = new string[] { "Mouse", "Duck", "Cat", "Mouse" };
    //Add collections to the marker variables where the name should match with
    input template
    marker.AddVariable("Names", names);
    marker.AddVariable("Descriptions", descriptions);
    //Process the markers in the template
    marker.ApplyMarkers();
    //Saving the workbook as stream
    workbook.Version = ExcelVersion.Excel2013;
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
    //Save the stream as Excel document and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    await
    DependencyService.Get<ISaveWindowsPhone>().SaveAndView("TemplateMarker.xlsx"
    , "application/msexcel", outputStream);
    else
    DependencyService.Get<ISave>().SaveAndView("TemplateMarker.xlsx",
    "application/msexcel", outputStream);
    //Dispose the input and output stream instances
}

```

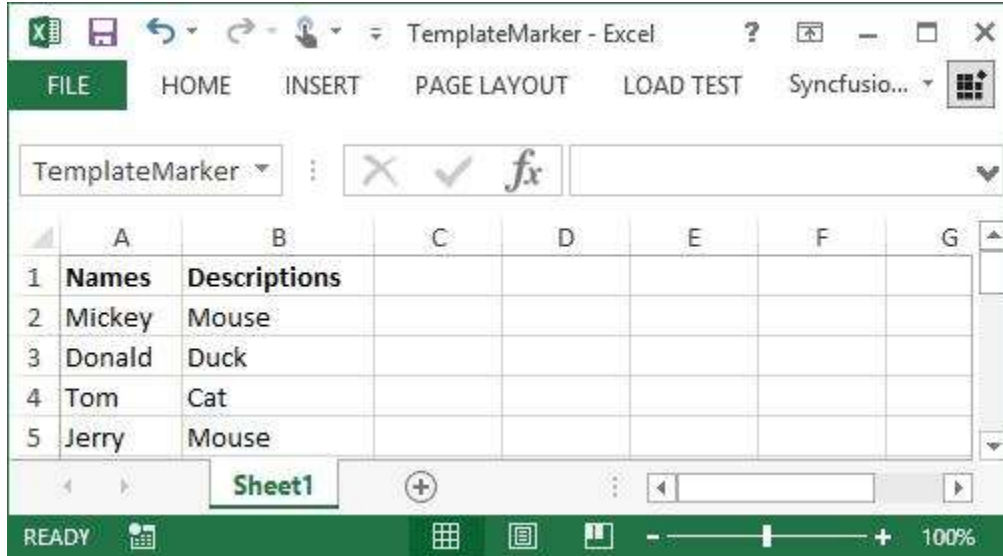


```

InputStream.Dispose();
OutputStream.Dispose();
}

```

The following screenshot represents generated Excel file in which the array of data is bounded.



You can also add or insert template markers using XlsIO APIs as follows.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IWorkbook workbook = excelEngine.Excel.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Insert Simple marker
    sheet.Range["B2"].Text = "%Marker";
    //Insert marker which gets value of Author name
    sheet.Range["C2"].Text = "%Marker2.Worksheet.Workbook.Author";
    //Insert marker which gets cell address
    sheet.Range["H2"].Text = "%ArrayProperty.Cells.Address";
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
        workbook.CreateTemplateMarkersProcessor();
    //Add collections to the marker variables where the name should match with
    input template
    marker.AddVariable("Marker", "First test of markers");
    marker.AddVariable("Marker2", sheet.Range["B2"]);
    marker.AddVariable("ArrayProperty", sheet.Range["B2:G2"]);
    //Process the markers in the template
    marker.ApplyMarkers();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs("TemplateMarker.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()

```

```

Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Open("Sample.xlsx")
IWorksheet sheet = workbook.Worksheets(0)
'Insert Simple marker
sheet.Range("B2").Text = "%Marker"
'Insert marker which gets value of Author name
sheet.Range("C2").Text = "%Marker2.Worksheet.Workbook.Author"
'Insert marker which gets cell address
sheet.Range("H2").Text = "%ArrayProperty.Cells.Address"
'Create Template Marker Processor
Dim marker As ITemplateMarkersProcessor =
workbook.CreateTemplateMarkersProcessor()
'Add collections to the marker variables where the name should match with
input template
marker.AddVariable("Marker", "First test of markers")
marker.AddVariable("Marker2", sheet.Range("B2"))
marker.AddVariable("ArrayProperty", sheet.Range("B2:G2"))
'Process the markers in the template
marker.ApplyMarkers()
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("TemplateMarker.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream = assembly.GetManifestResourceStream("Sample.xlsx");
    IWorkbook workbook = await
excelEngine.Excel.Workbooks.OpenAsync(inputStream);
IWorksheet sheet = workbook.Worksheets[0];
//Insert Simple marker
sheet.Range["B2"].Text = "%Marker";
//Insert marker which gets value of Author name
sheet.Range["C2"].Text = "%Marker2.Worksheet.Workbook.Author";
//Insert marker which gets cell address
sheet.Range["H2"].Text = "%ArrayProperty.Cells.Address";
//Create Template Marker Processor
ITemplateMarkersProcessor marker =
workbook.CreateTemplateMarkersProcessor();
//Add collections to the marker variables where the name should match with
input template
marker.AddVariable("Marker", "First test of markers");
marker.AddVariable("Marker2", sheet.Range["B2"]);
marker.AddVariable("ArrayProperty", sheet.Range["B2:G2"]);
//Process the markers in the template
marker.ApplyMarkers();
workbook.Version = ExcelVersion.Excel2013;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "TemplateMarker";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
}

```

```
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet sheet = workbook.Worksheets[0];
    //Insert Simple marker
    sheet.Range["B2"].Text = "%Marker";
    //Insert marker which gets value of Author name
    sheet.Range["C2"].Text = "%Marker2.Worksheet.Workbook.Author";
    //Insert marker which gets cell address
    sheet.Range["H2"].Text = "%ArrayProperty.Cells.Address";
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
    workbook.CreateTemplateMarkersProcessor();
    //Add collections to the marker variables where the name should match with
    input template
    marker.AddVariable("Marker", "First test of markers");
    marker.AddVariable("Marker2", sheet.Range["B2"]);
    marker.AddVariable("ArrayProperty", sheet.Range["B2:G2"]);
    //Process the markers in the template
    marker.ApplyMarkers();
    //Saving the workbook as stream
    workbook.Version = ExcelVersion.Excel2013;
    FileStream stream = new FileStream("TemplateMarker.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using(ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream = assembly.GetManifestResourceStream("Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    //Insert Simple marker
    sheet.Range["B2"].Text = "%Marker";
    //Insert marker which gets value of Author name
    sheet.Range["C2"].Text = "%Marker2.Worksheet.Workbook.Author";
    //Insert marker which gets cell address
    sheet.Range["H2"].Text = "%ArrayProperty.Cells.Address";
}
```

```

//Create Template Marker Processor
ITemplateMarkersProcessor marker =
workbook.CreateTemplateMarkersProcessor();
//Add collections to the marker variables where the name should match with
input template
marker.AddVariable("Marker", "First test of markers");
marker.AddVariable("Marker2", sheet.Range["B2"]);
marker.AddVariable("ArrayProperty", sheet.Range["B2:G2"]);
//Process the markers in the template
marker.ApplyMarkers();
//Saving the workbook as stream
workbook.Version = ExcelVersion.Excel2013;
MemoryStream outputStream = new MemoryStream();
workbook.SaveAs(outputStream);
//Save the stream as Excel document and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
await
DependencyService.Get<ISaveWindowsPhone>().SaveAndView("TemplateMarker.xlsx"
, "application/msexcel", outputStream);
else
DependencyService.Get<ISave>().SaveAndView("TemplateMarker.xlsx",
"application/msexcel", outputStream);
//Dispose the input and output stream instances
inputStream.Dispose();
outputStream.Dispose();
}

```

Bind from DataTable

Syntax:

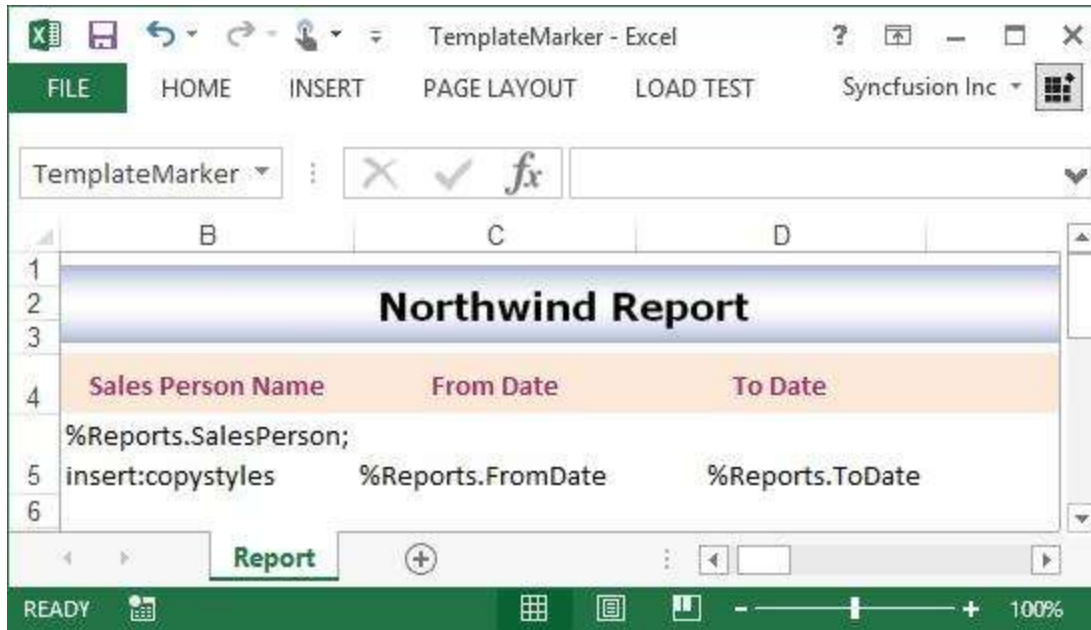
```
%<DataSource>.<FieldName>
```

For example: %Products.ProductName

Where, “Products” is a data source which can be data tables, datasets, data readers and data views and ProductName is the field name or column name

By default, DataTable values will be filled in the worksheet as a string format. You can detect data type and number format of DataTable values by using VariableTypeAction enumerator. To know more about the VariableTypeAction enumerator, please refer **VariableTypeAction** in API section.

The following screenshot represents the input template which has a template marker.



The following code snippet illustrates how to detect data type and apply number format with template marker.

C#

```
using(ExcelEngine excelEngine = new ExcelEngine())
{
    IWorkbook workbook =
    excelEngine.Excel.Workbooks.Open("TemplateMarker.xlsx");
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
    workbook.CreateTemplateMarkersProcessor();
    DataTable reports = new DataTable();
    reports.Columns.Add("SalesPerson");
    reports.Columns.Add("FromDate", typeof(DateTime));
    reports.Columns.Add("ToDate", typeof(DateTime));
    reports.Rows.Add("Andy Bernard", new DateTime(2014, 09, 08), new
    DateTime(2014, 09, 11));
    reports.Rows.Add("Jim Halpert", new DateTime(2014, 09, 11), new
    DateTime(2014, 09, 15));
    reports.Rows.Add("Karen Fillippelli", new DateTime(2014, 09, 15), new
    DateTime(2014, 09, 20));
    reports.Rows.Add("Phyllis Lapin", new DateTime(2014, 09, 21), new
    DateTime(2014, 09, 25));
    reports.Rows.Add("Stanley Hudson", new DateTime(2014, 09, 26), new
    DateTime(2014, 09, 30));
    //Add collection to the marker variables where the name should match with
    input template
    //Detects number format in DataTable values
    marker.AddVariable("Reports",
    reports, VariableTypeAction.DetectNumberFormat);
    //Process the markers and detect the number format along with the data type
    in the template
    marker.ApplyMarkers();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs("TemplateMarkerWithFormat.xlsx");
}
```

```
}

```

VB.NET

```
using excelEngine As ExcelEngine = new ExcelEngine()
Dim workbook As IWorkbook =
excelEngine.Excel.Workbooks.Open("TemplateMarker.xlsx")
'Create Template Marker Processor
Dim marker As ITemplateMarkersProcessor =
workbook.CreateTemplateMarkersProcessor()
Dim reports As New DataTable()
reports.Columns.Add("SalesPerson")
reports.Columns.Add("FromDate", GetType(DateTime))
reports.Columns.Add("ToDate", GetType(DateTime))
reports.Rows.Add("Andy Bernard", New DateTime(2014, 9, 8), New
DateTime(2014, 9, 11))
reports.Rows.Add("Jim Halpert", New DateTime(2014, 9, 11), New
DateTime(2014, 9, 15))
reports.Rows.Add("Karen Fillippelli", New DateTime(2014, 9, 15), New
DateTime(2014, 9, 20))
reports.Rows.Add("Phyllis Lapin", New DateTime(2014, 9, 21), New
DateTime(2014, 9, 25))
reports.Rows.Add("Stanley Hudson", New DateTime(2014, 9, 26), New
DateTime(2014, 9, 30))
'Add collection to the marker variables where the name should match with
input template
'Detects number format in DataTable values
marker.AddVariable("Reports", reports,
VariableTypeAction.DetectNumberFormat)
'Process the markers and detect the number format along with the data type
in the template
marker.ApplyMarkers()
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("TemplateMarkerWithFormat.xlsx")
End Using
```

UWP

```
//XlsIO supports binding data from data table using template markers in
Windows Forms, WPF, ASP.NET, ASP.NET MVC, and ASP.NET Core (2.0 onwards)
platforms alone.
```

ASP.NET CORE

```
//Binding data from data table is supported only from ASP.NET Core 2.0
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
FileStream fileStream = new FileStream("TemplateMarker.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream);
//Create Template Marker Processor
ITemplateMarkersProcessor marker =
workbook.CreateTemplateMarkersProcessor();
DataTable reports = new DataTable();
```

```

reports.Columns.Add("SalesPerson");
reports.Columns.Add("FromDate", typeof(DateTime));
reports.Columns.Add("ToDate", typeof(DateTime));
reports.Rows.Add("Andy Bernard", new DateTime(2014, 09, 08), new
DateTime(2014, 09, 11));
reports.Rows.Add("Jim Halpert", new DateTime(2014, 09, 11), new
DateTime(2014, 09, 15));
reports.Rows.Add("Karen Fillippelli", new DateTime(2014, 09, 15), new
DateTime(2014, 09, 20));
reports.Rows.Add("Phyllis Lapin", new DateTime(2014, 09, 21), new
DateTime(2014, 09, 25));
reports.Rows.Add("Stanley Hudson", new DateTime(2014, 09, 26), new
DateTime(2014, 09, 30));
//Add collection to the marker variables where the name should match with
input template
//Detects number format in DataTable values
marker.AddVariable("Reports", reports,
VariableTypeAction.DetectNumberFormat);
//Process the markers and detect the number format along with the data type
in the template
marker.ApplyMarkers();
//Saving the workbook as stream
workbook.Version = ExcelVersion.Excel2013;
FileStream stream = new FileStream("TemplateMarkerWithFormat.xlsx",
FileMode.Create, FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

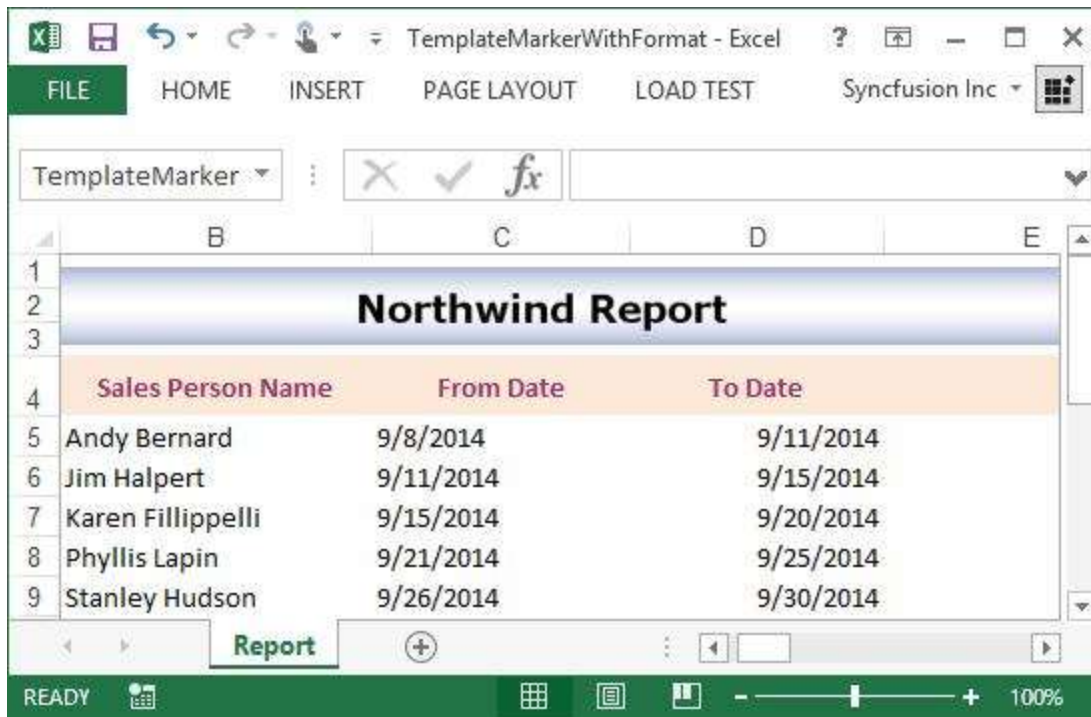
XAMARIN

```

//XlsIO supports binding data from data table using template markers in
Windows Forms, WPF, ASP.NET, ASP.NET MVC, and ASP.NET Core (2.0 onwards)
platforms alone.

```

The following screenshot represents an Excel file in which the data type is detected and then number format is applied.



Bind from Collection Objects with images

You can generate reports more appealingly with image support in template markers. The possible image formats are as follows:

- GIF
- JPEG
- PNG
- BMP
- TIFF

XlsIO detects the property as image when its type is `System.Drawing.Image` or `byte []`. The image can be formatted using the following arguments.

No	Image arguments	Description
1	No argument Ex: %Reports.Image;	Image is applied with a default size (50x50 pixels) and position (Top-Left).
2	fittocell Ex: %Reports.Image;fittocell	The image is applied to cell width and height.
3	size:width,height Ex: %Reports.Image;size:60	Image is applied to the specified size (width, height). Height parameter is optional. Value of width is applied when height is not specified.

	(or) <code>%Reports.Image;size:60,60</code> (or) <code>%Reports.Image;size:60,auto</code> (or) <code>%Reports.Image;size:auto,60</code> (or) <code>%Reports.Image;size:auto,auto</code>	You can specify either width or height value as "auto" to set ratio value of other size value. If both width and height value is set as "auto" then the Image is applied to the original size
4	<code>position:position</code> Ex: <code>%Reports.Image;position:middle-center</code> (or) <code>%Reports.Image;position:right</code>	Image is positioned (top-left, top-center, etc.,) within the cell.

In the following example, a marker is added for merging images. Data source and property name is specified (`%Reports.Image;`) for image also.

Sales Person Photo	Name	Sales Jan- June	Sales July - Dec	Change
<code>%Reports.Image;insert:copystyles</code>	<code>%Reports.SalesPerson</code>	<code>%Reports.SalesJanJun</code>	<code>%Reports.SalesJulDec</code>	<code>%Reports.Change</code>

Marker added for merging images

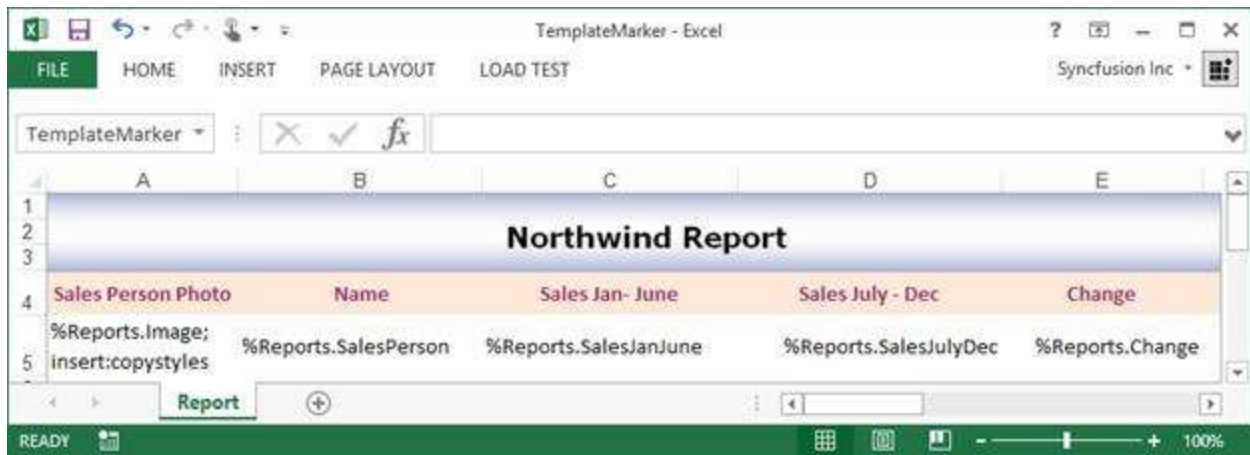
Note: Image can be used in array, DataTable, and collection objects.

Different positions of the image are maintained internally in the `ImageVerticalPosition` and `ImageHorizontalPosition` enumerators. To learn more about this, refer to the **ImageVerticalPosition** and **ImageHorizontalPosition** enumerators respectively in API section.

The output of all the image insertion options with input templates are as follows.

Default image input and output

Input template



Generated output

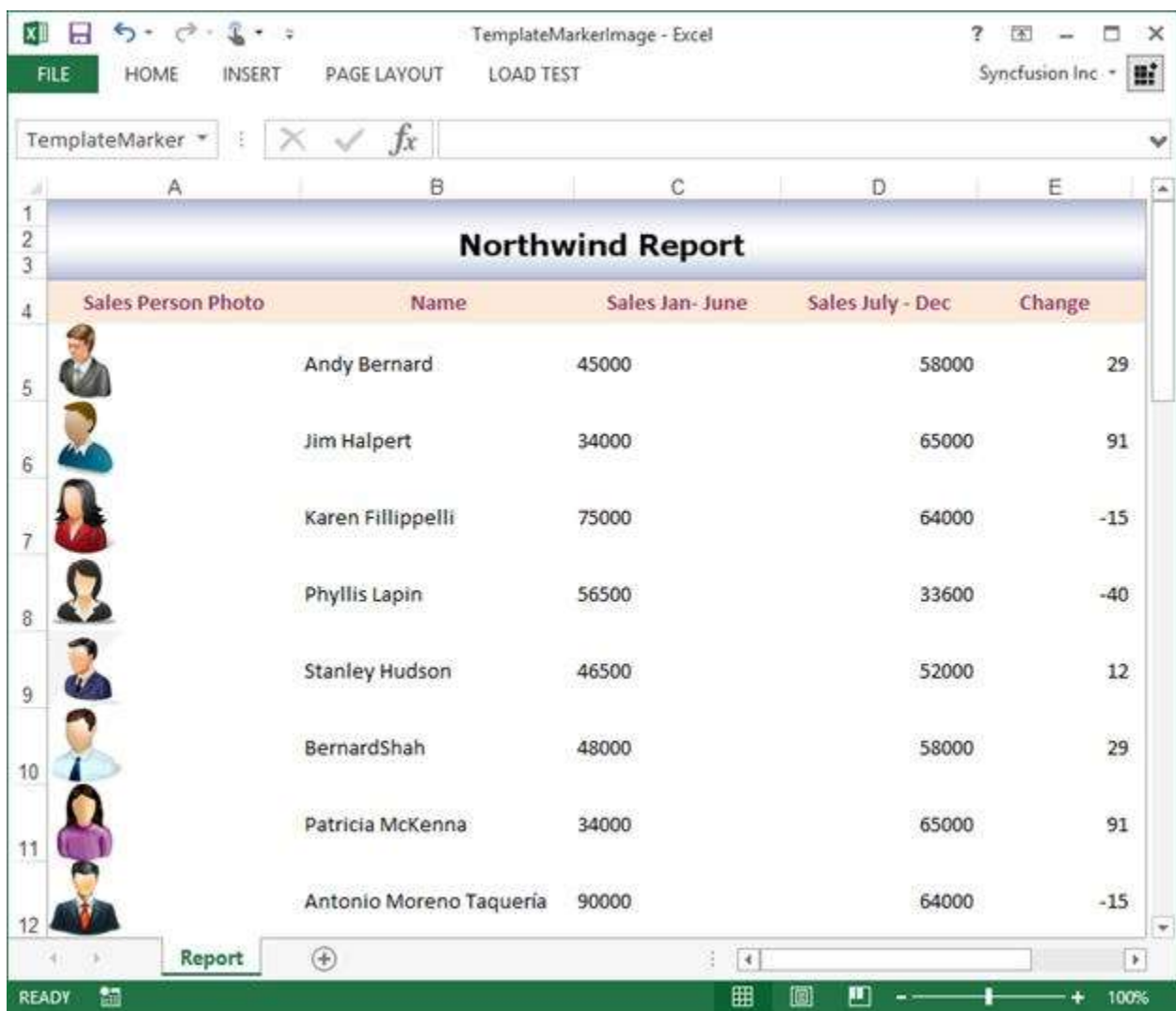
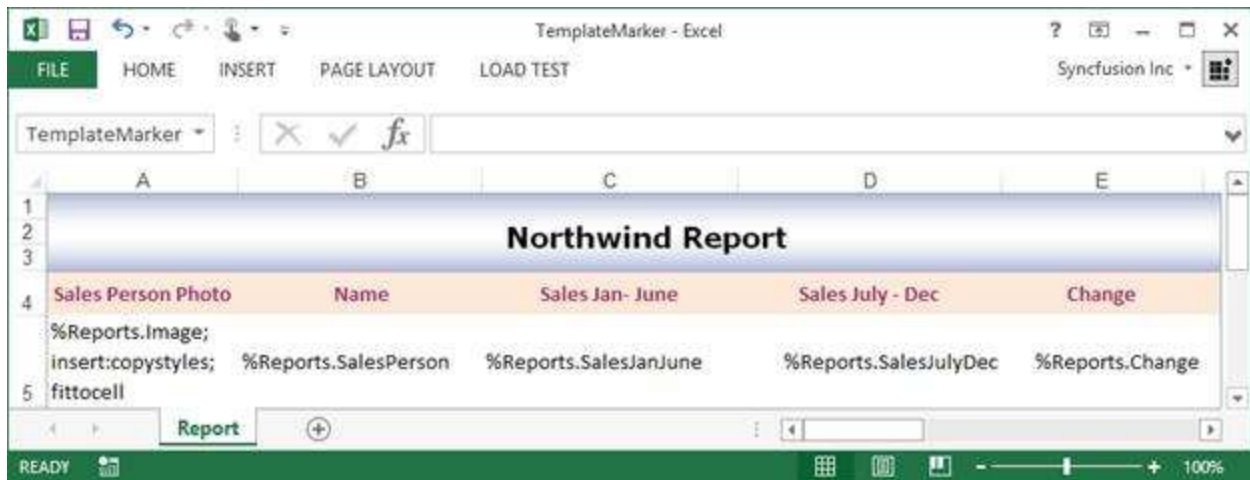


Image with FitToCell attribute

Input template



Generated output

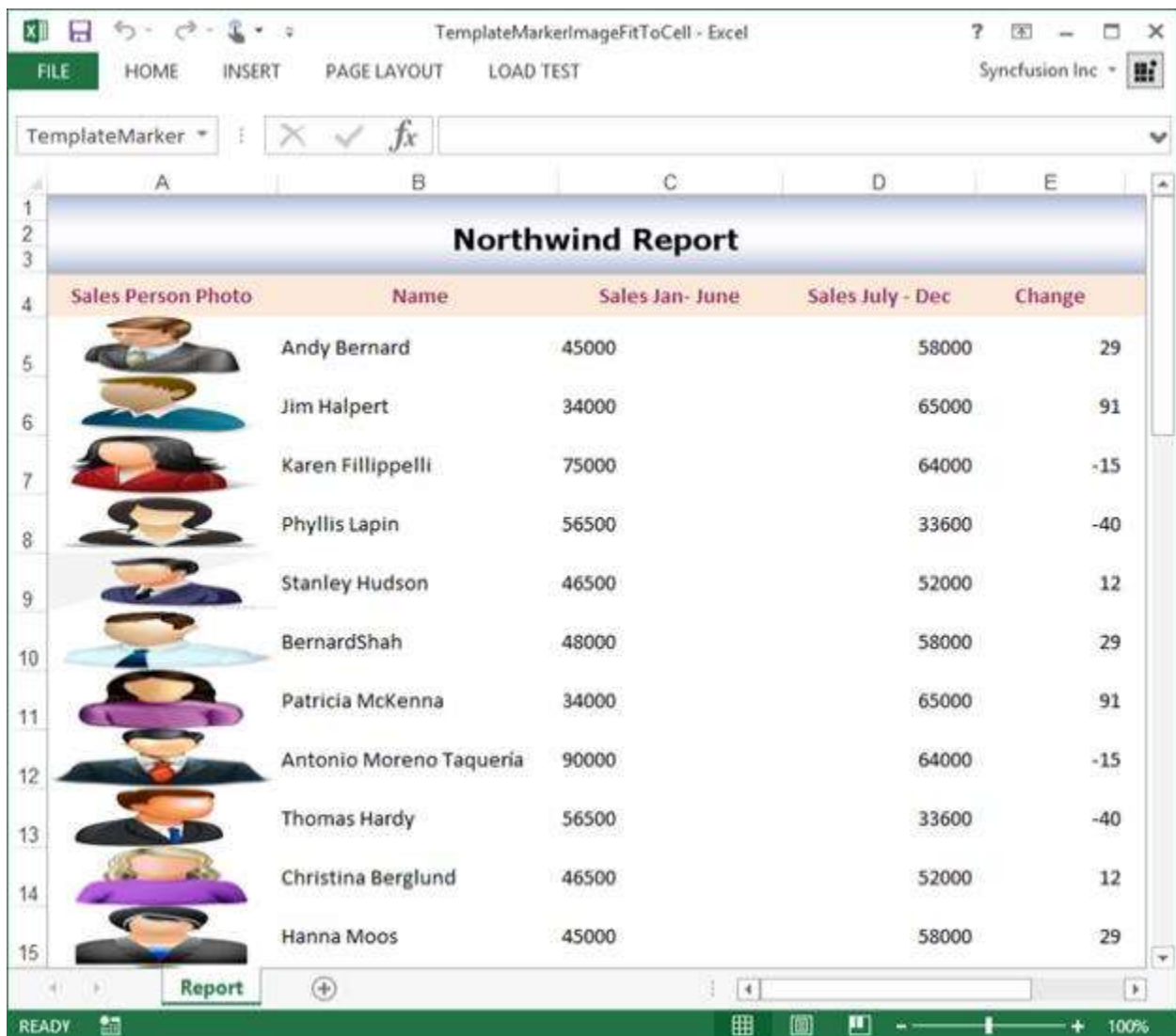
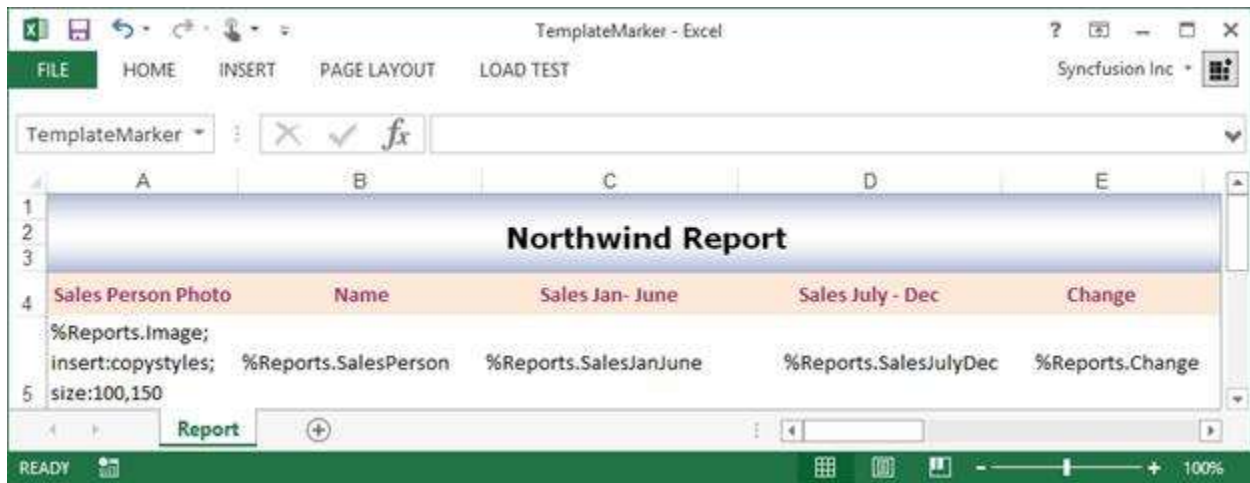


Image with Size

Input template



Generated output

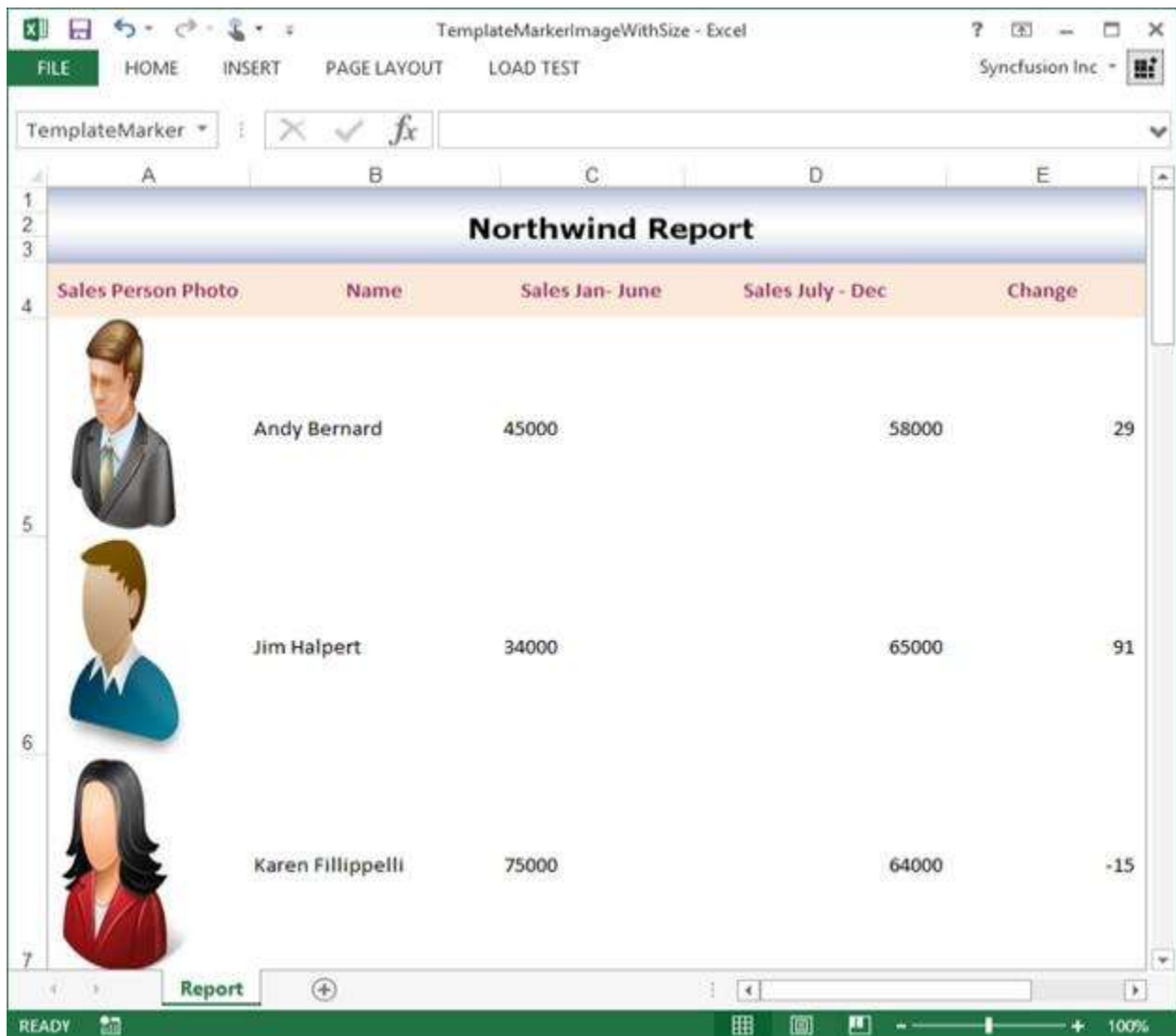
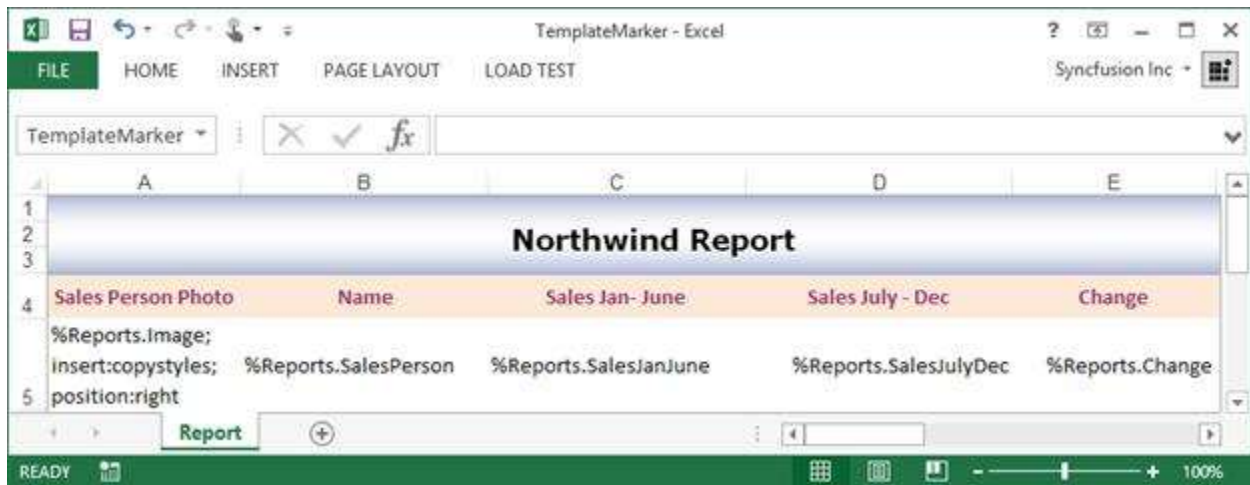


Image with Position

Input template



Generated output

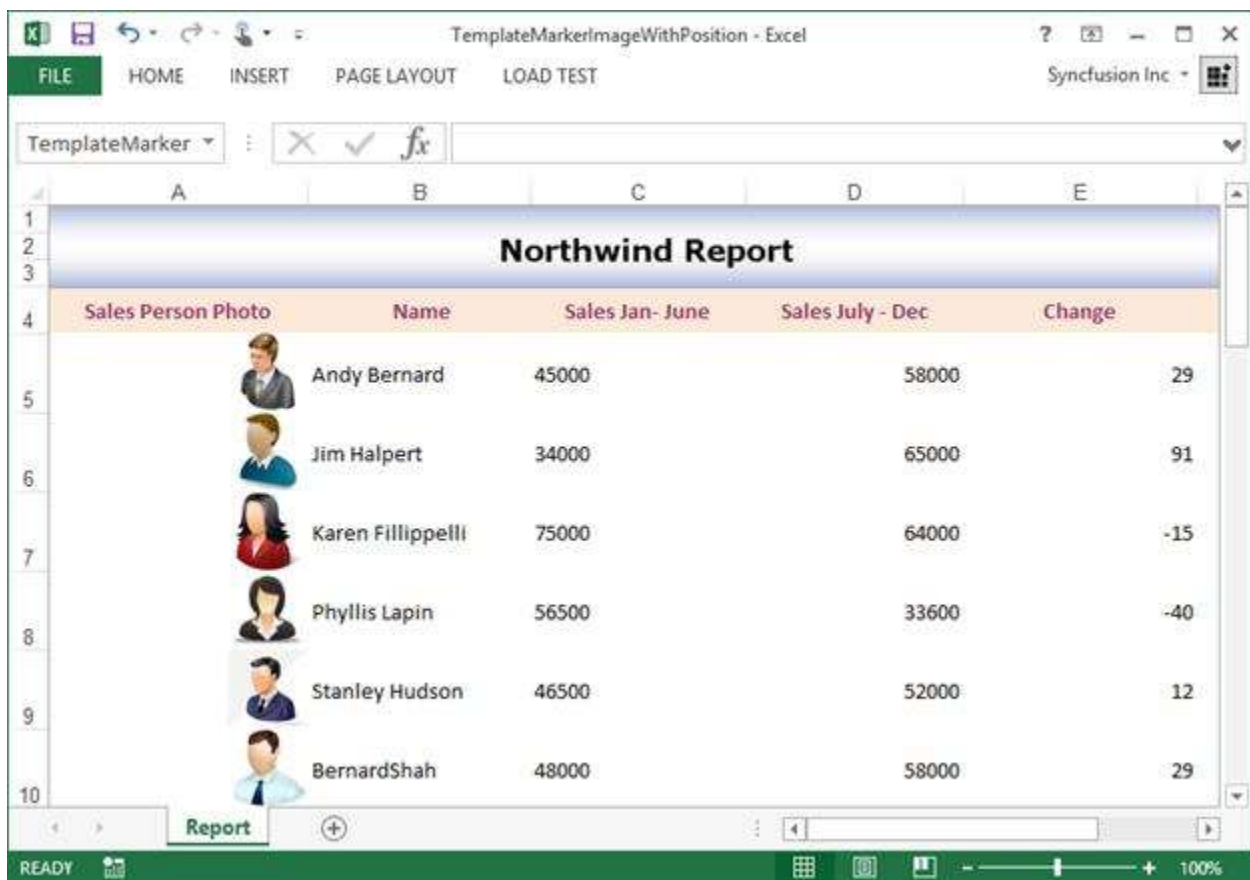
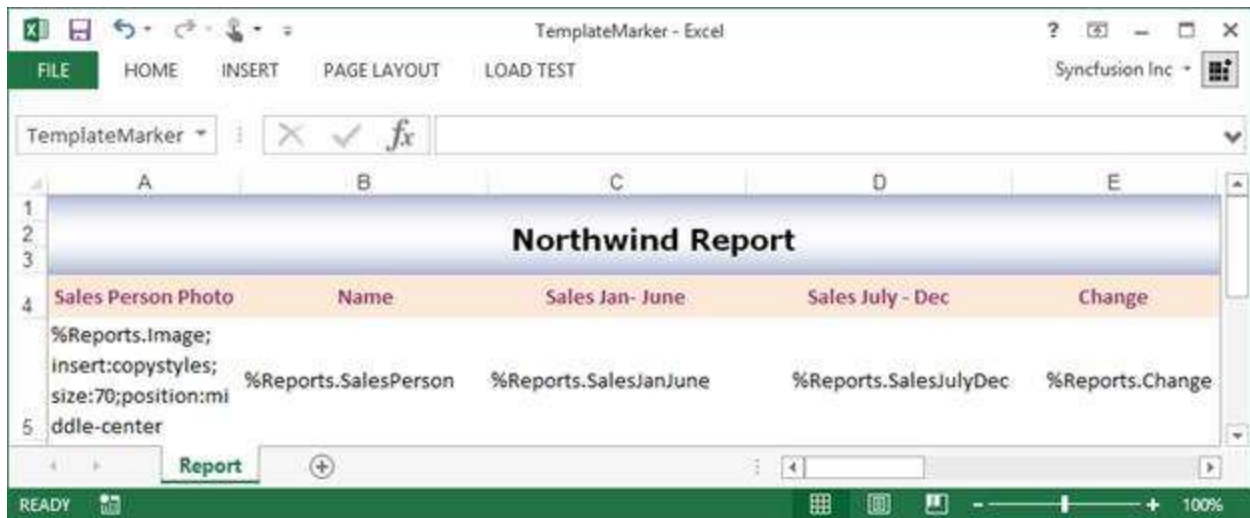
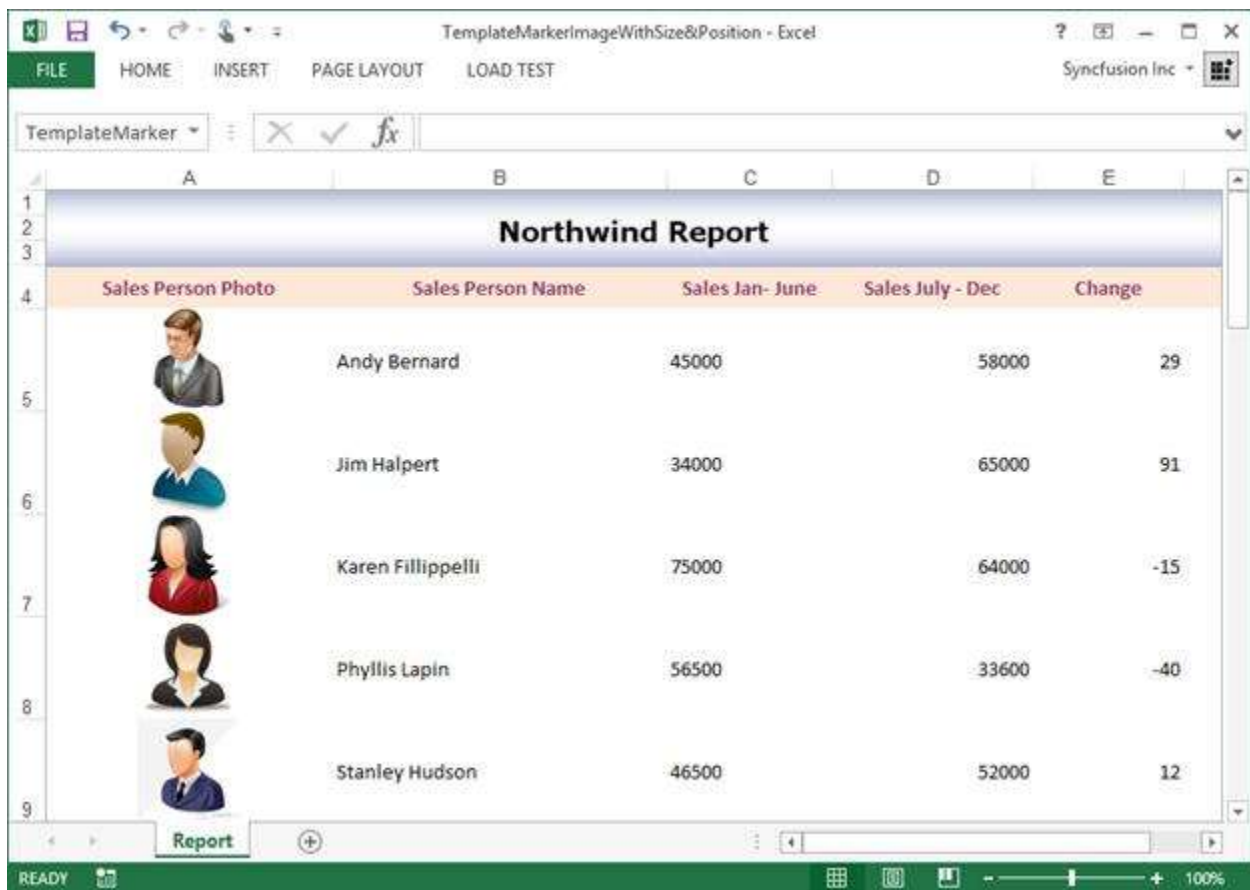


Image with position and size

Input template



Generated output



You can also refer to the [Template based data filling using Template Markers](#) section in [Getting Started](#) for the sample regarding template marker with images.

Bind from Nested Collection Objects with import data and group options

You can bind the data from Nested collection objects with import data options and group options.

The import data options are:

- Default
- Merge
- Repeat

The import data group options are:

- Collapse Group
- Expand Group

The following code snippet illustrates how to import data from nested collection objects with template marker.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("TemplateMarker.xlsx")
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
    workbook.CreateTemplateMarkersProcessor();
    //Add collection to the marker variables where the name should match with
    input template
    marker.AddVariable("Customer", GetSalesReports());
    //Process the markers in the template
    marker.ApplyMarkers();
    workbook.SaveAs("TemplateMarkerNestedCollection.xlsx");
}
```

VB.NET

```
using excelEngine As ExcelEngine = new ExcelEngine()
Dim workbook As IWorkbook =
excelEngine.Excel.Workbooks.Open("TemplateMarker.xlsx")
'Create Template Marker Processor
Dim marker As ITemplateMarkersProcessor =
workbook.CreateTemplateMarkersProcessor()
'Add collection to the marker variables where the name should match with
input template
marker.AddVariable("Customer", GetSalesReports())
'Process the markers and detect the number format along with the data type
in the template
marker.ApplyMarkers()
workbook.SaveAs("TemplateMarkerNestedCollection.xlsx");
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
```

```

openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile openFile = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(openFile);
//Create Template Marker Processor
ITemplateMarkersProcessor marker =
workbook.CreateTemplateMarkersProcessor();
//Add collection to the marker variables where the name should match with
input template
marker.AddVariable("Customer", GetSalesReports());
//Process the markers in the template
marker.ApplyMarkers();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "TemplateMarkerNestedCollection";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

//Binding data from data table is supported only from ASP.NET Core 2.0
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
FileStream fileStream = new FileStream("TemplateMarker.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream);
//Create Template Marker Processor
ITemplateMarkersProcessor marker =
workbook.CreateTemplateMarkersProcessor();
//Add collection to the marker variables where the name should match with
input template
marker.AddVariable("Customer", GetSalesReports());
//Process the markers in the template
marker.ApplyMarkers();
//Saving the workbook as stream
FileStream stream = new FileStream("TemplateMarkerNestedCollection.xlsx",
FileMode.Create, FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
Assembly assembly = typeof(App).GetTypeInfo().Assembly;

```



```

Stream inputStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.TemplateMarker.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream);
//Create Template Marker Processor
ITemplateMarkersProcessor marker =
workbook.CreateTemplateMarkersProcessor();
//Add collection to the marker variables where the name should match with
input template
marker.AddVariable("Customer", GetSalesReports());
//Process the markers in the template
marker.ApplyMarkers();
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("TemplateMarkerNestedCollection.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TemplateMarkerNestedCollection.xlsx", "application/msexcel", stream);
}
}

```

The following code snippet provides supporting methods and classes for the previous code.

C#

```

//Gets a list of sales reports
public static List<Customer> GetSalesReports()
{
List<Customer> reports = new List<Customer>();
List<Order> orders = new List<Order>();
orders.Add(new Order(1408, 451.75));
orders.Add(new Order(1278, 340.00));
orders.Add(new Order(1123, 290.50));
Customer c1 = new Customer(002107, "Andy Bernard", 45);
c1.Orders = orders;
Customer c2 = new Customer(011564, "Jim Halpert", 34);
c2.Orders = orders;
Customer c3 = new Customer(002097, "Karen Fillippelli", 35);
c3.Orders = orders;
Customer c4 = new Customer(001846, "Phyllis Lapin", 37);
c4.Orders = orders;
Customer c5 = new Customer(012167, "Stanley Hudson", 41);
c5.Orders = orders;
reports.Add(c1);
}

```

```

reports.Add(c2);
reports.Add(c3);
reports.Add(c4);
reports.Add(c5);
return reports;
}
//Customer details
public partial class Customer
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Age { get; set; }
    public IList<Order> Orders { get; set; }
    public Customer(int id, string name, int age)
    {
        Id = id;
        Name = name;
        Age = age;
    }
}
//Order details
public partial class Order
{
    public int Order_Id { get; set; }
    public double Price { get; set; }
    public Order(int id, double price)
    {
        Order_Id = id;
        Price = price;
    }
}

```

VB.NET

```

'Gets a list of sales reports
Public Shared Function GetSalesReports() As List(Of Customer)
Dim reports As List(Of Customer) = New List(Of Customer) ()
Dim orders As List(Of Order) = New List(Of Order) ()
orders.Add(New Order(1408, 451.75))
orders.Add(New Order(1278, 340.00))
orders.Add(New Order(1123, 290.50))
Dim c1 As Customer = New Customer(002107, "Andy Bernard", 45)
c1.Orders = orders
Dim c2 As Customer = New Customer(011564, "Jim Halpert", 34)
c2.Orders = orders
Dim c3 As Customer = New Customer(002097, "Karen Fillippelli", 35)
c3.Orders = orders
Dim c4 As Customer = New Customer(001846, "Phyllis Lapin", 37)
c4.Orders = orders
Dim c5 As Customer = New Customer(012167, "Stanley Hudson", 41)
c5.Orders = orders
reports.Add(c1)
reports.Add(c2)
reports.Add(c3)
reports.Add(c4)
reports.Add(c5)

```

```

Return reports
End Function
'Customer details
Public Class Customer
Public Property Id As Integer
Public Property Name As String
Public Property Age As Integer
Public Property Orders As IList(Of Order)
Public Sub New(ByVal id As Integer, ByVal name As String, ByVal age As
Integer)
Id = id
Name = name
Age = age
End Sub
End Class
'Order details
Public Class Order
Public Property Order_Id As Integer
Public Property Price As Double
Public Sub New(ByVal id As Integer, ByVal price As Double)
Order_Id = id
Price = price
End Sub
End Class

```

UWP

```

//Gets a list of sales reports
public static List<Customer> GetSalesReports()
{
List<Customer> reports = new List<Customer>();
List<Order> orders = new List<Order>();
orders.Add(new Order(1408, 451.75));
orders.Add(new Order(1278, 340.00));
orders.Add(new Order(1123, 290.50));
Customer c1 = new Customer(002107, "Andy Bernard", 45);
c1.Orders = orders;
Customer c2 = new Customer(011564, "Jim Halpert", 34);
c2.Orders = orders;
Customer c3 = new Customer(002097, "Karen Fillippelli", 35);
c3.Orders = orders;
Customer c4 = new Customer(001846, "Phyllis Lapin", 37);
c4.Orders = orders;
Customer c5 = new Customer(012167, "Stanley Hudson", 41);
c5.Orders = orders;
reports.Add(c1);
reports.Add(c2);
reports.Add(c3);
reports.Add(c4);
reports.Add(c5);
return reports;
}
//Customer details
public partial class Customer
{
public int Id { get; set; }
}

```

```

public string Name { get; set; }
public int Age { get; set; }
public IList<Order> Orders { get; set; }
public Customer(int id, string name, int age)
{
    Id = id;
    Name = name;
    Age = age;
}
}
//Order details
public partial class Order
{
    public int Order_Id { get; set; }
    public double Price { get; set; }
    public Order(int id, double price)
    {
        Order_Id = id;
        Price = price;
    }
}

```

ASP.NET CORE

```

//Gets a list of sales reports
public static List<Customer> GetSalesReports()
{
    List<Customer> reports = new List<Customer>();
    List<Order> orders = new List<Order>();
    orders.Add(new Order(1408, 451.75));
    orders.Add(new Order(1278, 340.00));
    orders.Add(new Order(1123, 290.50));
    Customer c1 = new Customer(002107, "Andy Bernard", 45);
    c1.Orders = orders;
    Customer c2 = new Customer(011564, "Jim Halpert", 34);
    c2.Orders = orders;
    Customer c3 = new Customer(002097, "Karen Fillippelli", 35);
    c3.Orders = orders;
    Customer c4 = new Customer(001846, "Phyllis Lapin", 37);
    c4.Orders = orders;
    Customer c5 = new Customer(012167, "Stanley Hudson", 41);
    c5.Orders = orders;
    reports.Add(c1);
    reports.Add(c2);
    reports.Add(c3);
    reports.Add(c4);
    reports.Add(c5);
    return reports;
}
//Customer details
public partial class Customer
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Age { get; set; }
    public IList<Order> Orders { get; set; }
}

```

```

public Customer(int id, string name, int age)
{
    Id = id;
    Name = name;
    Age = age;
}
}
//Order details
public partial class Order
{
    public int Order_Id { get; set; }
    public double Price { get; set; }
    public Order(int id, double price)
    {
        Order_Id = id;
        Price = price;
    }
}
}

```

XAMARIN

```

//Gets a list of sales reports
public static List<Customer> GetSalesReports()
{
    List<Customer> reports = new List<Customer>();
    List<Order> orders = new List<Order>();
    orders.Add(new Order(1408, 451.75));
    orders.Add(new Order(1278, 340.00));
    orders.Add(new Order(1123, 290.50));
    Customer c1 = new Customer(002107, "Andy Bernard", 45);
    c1.Orders = orders;
    Customer c2 = new Customer(011564, "Jim Halpert", 34);
    c2.Orders = orders;
    Customer c3 = new Customer(002097, "Karen Fillippelli", 35);
    c3.Orders = orders;
    Customer c4 = new Customer(001846, "Phyllis Lapin", 37);
    c4.Orders = orders;
    Customer c5 = new Customer(012167, "Stanley Hudson", 41);
    c5.Orders = orders;
    reports.Add(c1);
    reports.Add(c2);
    reports.Add(c3);
    reports.Add(c4);
    reports.Add(c5);
    return reports;
}
//Customer details
public partial class Customer
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Age { get; set; }
    public IList<Order> Orders { get; set; }
    public Customer(int id, string name, int age)
    {
        Id = id;
    }
}

```

```

Name = name;
Age = age;
}
}
//Order details
public partial class Order
{
    public int Order_Id { get; set; }
    public double Price { get; set; }
    public Order(int id, double price)
    {
        Order_Id = id;
        Price = price;
    }
}

```

The output of all the import data and group options with input templates are as follows.

Default option input and output

Input template

Sales Report				
Customer Name	Customer Id	Customer Age	Order Id	Order Price
%Customer.Name	%Customer.Id	%Customer.Age	%Customer.Orders.Order_Id	%Customer.Orders.Price

Generated output

AutoSave TemplateMarker_Default... Sign in

File Home Insert Data Tell me what you want to do Share Comments

TemplateMarker

	A	B	C	D	E
1					
2	Sales Report				
3					
4	Customer Name	Customer Id	Customer Age	Order Id	Order Price
5	Andy Bernard	2107	45	1408	451.75
6				1278	340
7				1123	290.5
8	Jim Halpert	11564	34	1408	451.75
9				1278	340
10				1123	290.5
11	Karen Filippelli	2097	35	1408	451.75
12				1278	340
13				1123	290.5
14	Phyllis Lapin	1846	37	1408	451.75
15				1278	340
16				1123	290.5
17	Stanley Hudson	12167	41	1408	451.75
18				1278	340
19				1123	290.5

Sheet1 100%

Merge option input and output

Input template

The screenshot shows an Excel spreadsheet with a 'Sales Report' template. The report is structured as follows:

Customer Name	Customer Id	Customer Age	Order Id	Order Price
%Customer.Name; merge	%Customer.Id; merge	%Customer.Age; merge	%Customer.Orders.Order_Id	%Customer.Orders.Price

Generated output

The screenshot shows an Excel spreadsheet with a 'Sales Report' table. The table has 5 columns: Customer Name, Customer Id, Customer Age, Order Id, and Order Price. The data is organized into groups of 3 rows for each customer, with the first row of each group containing the customer's details and the subsequent two rows containing order details. The first customer, Andy Bernard, has a Customer Id of 2107 and Age of 45, with three orders (1408, 1278, 1123). This pattern repeats for five other customers: Jim Halpert, Karen Fillippelli, Phyllis Lapin, and Stanley Hudson. The bottom of the spreadsheet shows 'Sheet1' as the active sheet and a zoom level of 100%.

	Customer Name	Customer Id	Customer Age	Order Id	Order Price
5	Andy Bernard	2107	45	1408	451.75
6				1278	340
7				1123	290.5
8	Jim Halpert	11564	34	1408	451.75
9				1278	340
10				1123	290.5
11	Karen Fillippelli	2097	35	1408	451.75
12				1278	340
13				1123	290.5
14	Phyllis Lapin	1846	37	1408	451.75
15				1278	340
16				1123	290.5
17	Stanley Hudson	12167	41	1408	451.75
18				1278	340
19				1123	290.5

Repeat option input and output

Input template

The screenshot shows an Excel spreadsheet with a 'Sales Report' template. The report is structured as follows:

Customer Name	Customer Id	Customer Age	Order Id	Order Price
%Customer.Name; repeat	%Customer.Id; repeat	%Customer.Age; repeat	%Customer.Orders.Order_Id	%Customer.Orders.Price

Generated output

The screenshot shows an Excel spreadsheet with a 'Sales Report' table. The table has five columns: Customer Name, Customer Id, Customer Age, Order Id, and Order Price. The data is organized into groups of three rows each, with the first row of each group having a highlighted 'Customer Name' cell. The formula bar shows 'Andy Bernard'.

	Customer Name	Customer Id	Customer Age	Order Id	Order Price
5	Andy Bernard	2107	45	1408	451.75
6	Andy Bernard	2107	45	1278	340
7	Andy Bernard	2107	45	1123	290.5
8	Jim Halpert	11564	34	1408	451.75
9	Jim Halpert	11564	34	1278	340
10	Jim Halpert	11564	34	1123	290.5
11	Karen Fillippelli	2097	35	1408	451.75
12	Karen Fillippelli	2097	35	1278	340
13	Karen Fillippelli	2097	35	1123	290.5
14	Phyllis Lapin	1846	37	1408	451.75
15	Phyllis Lapin	1846	37	1278	340
16	Phyllis Lapin	1846	37	1123	290.5
17	Stanley Hudson	12167	41	1408	451.75
18	Stanley Hudson	12167	41	1278	340
19	Stanley Hudson	12167	41	1123	290.5

Collapse group option input and output

Input template

	A	B	C	D	E
1					
2					
3	Sales Report				
4	Customer Name	Customer Id	Customer Age	Order Id	Order Price
5	%Customer.Name; collapsegroup	%Customer.Id	%Customer.Age	%Customer.Orders.Order_Id	%Customer.Orders.Price
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

Generated output

The screenshot displays an Excel spreadsheet with a 'Sales Report' table. The table is structured as follows:

Customer Name	Customer Id	Customer Age	Order Id	Order Price
Andy Bernard	2107	45	1408	451.75
Jim Halpert	11564	34	1408	451.75
Karen Fillippelli	2097	35	1408	451.75
Phyllis Lapin	1846	37	1408	451.75
Stanley Hudson	12167	41	1408	451.75

Expand group option input and output

Input template

	A	B	C	D	E
1	Sales Report				
2					
3					
4	Customer Name	Customer Id	Customer Age	Order Id	Order Price
5	%Customer.Name; expandgroup	%Customer.Id	%Customer.Age	%Customer.Orders.Order_Id	%Customer.Orders.Price
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

Generated output

Sales Report				
Customer Name	Customer Id	Customer Age	Order Id	Order Price
Andy Bernard	2107	45	1408	451.75
			1278	340
			1123	290.5
Jim Halpert	11564	34	1408	451.75
			1278	340
			1123	290.5
Karen Filippelli	2097	35	1408	451.75
			1278	340
			1123	290.5
Phyllis Lapin	1846	37	1408	451.75
			1278	340
			1123	290.5
Stanley Hudson	12167	41	1408	451.75
			1278	340
			1123	290.5

Template marker with conditional formatting

You can create or apply conditional format to the template marker range.

The following screenshot represents the input template, which has a template marker.

Northwind Report			
Sales Person Name	Sales Jan- June	Sales July - Dec	Change
%Reports.SalesPerson;			
insert:copystyles	%Reports.SalesJanJun	%Reports.SalesJulDec	%Reports.Change

The following code sample illustrates how to create or apply conditional format to the marker.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    #region Initialize Workbook
    IWorkbook workbook =
    excelEngine.Excel.Workbooks.Open("TemplateMarker.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    #endregion
    #region Create Template Marker
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
    workbook.CreateTemplateMarkersProcessor();
    IConditionalFormats conditionalFormats =
    marker.CreateConditionalFormats(worksheet["C5"]);
    #region Data Bar
    //Apply markers using Formula
    IConditionalFormat condition = conditionalFormats.AddCondition();
    //Set Data bar and icon set for the same cell
    //Set the format type
    condition.FormatType = ExcelCFTType.DataBar;
    IDataBar dataBar = condition.DataBar;
    //Set the constraint
    dataBar.MinPoint.Type = ConditionValueType.LowestValue;
    dataBar.MinPoint.Value = "0";
    dataBar.MaxPoint.Type = ConditionValueType.HighestValue;
    dataBar.MaxPoint.Value = "0";
    //Set color for Bar
    dataBar.BarColor = Color.FromArgb(156, 208, 243);
    //Hide the value in data bar
    dataBar.ShowValue = false;
    #endregion
    #region IconSet
    condition = conditionalFormats.AddCondition();
    condition.FormatType = ExcelCFTType.IconSet;
    IIconSet iconSet = condition.IconSet;
    iconSet.IconSet = ExcelIconSetType.FourRating;
    iconSet.IconCriteria[0].Type = ConditionValueType.LowestValue;
    iconSet.IconCriteria[0].Value = "0";
    iconSet.IconCriteria[1].Type = ConditionValueType.HighestValue;
    iconSet.IconCriteria[1].Value = "0";
    iconSet.ShowIconOnly = true;
    #endregion
    conditionalFormats = marker.CreateConditionalFormats(worksheet["D5"]);
    #region Color Scale
    condition = conditionalFormats.AddCondition();
    condition.FormatType = ExcelCFTType.ColorScale;
    IColorScale colorScale = condition.ColorScale;
    //Sets 3 - color scale
    colorScale.SetConditionCount(3);
    colorScale.Criteria[0].FormatColorRGB = Color.FromArgb(230, 197, 218);
    colorScale.Criteria[0].Type = ConditionValueType.LowestValue;
    colorScale.Criteria[0].Value = "0";
    colorScale.Criteria[1].FormatColorRGB = Color.FromArgb(244, 210, 178);
    colorScale.Criteria[1].Type = ConditionValueType.Percentile;
```



```

colorScale.Criteria[1].Value = "50";
colorScale.Criteria[2].FormatColorRGB = Color.FromArgb(245, 247, 171);
colorScale.Criteria[2].Type = ConditionValueType.HighestValue;
colorScale.Criteria[2].Value = "0";
#endregion
conditionalFormats = marker.CreateConditionalFormats(worksheet["E5"]);
#region IconSet
condition = conditionalFormats.AddCondition();
condition.FormatType = ExcelCfType.IconSet;
iconSet = condition.IconSet;
iconSet.IconSet = ExcelIconSetType.ThreeSymbols;
iconSet.IconCriteria[0].Type = ConditionValueType.LowestValue;
iconSet.IconCriteria[0].Value = "0";
iconSet.IconCriteria[1].Type = ConditionValueType.HighestValue;
iconSet.IconCriteria[1].Value = "0";
iconSet.ShowIconOnly = false;
#endregion
//Add collection to the marker variables where the name should match with
input template
marker.AddVariable("Reports", GetSalesReports());
//Process the markers in the template
marker.ApplyMarkers();
#endregion
#region Save the Workbook
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("TemplateMarkerCF.xlsx");
#endregion
}

```

VB.NET

```

'Region "Initialize Workbook"
using excelEngine As ExcelEngine = new ExcelEngine()
Dim workbook As IWorkbook =
excelEngine.Excel.Workbooks.Open("TemplateMarker.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'End Region
'Region "Create Template Marker"
'Create Template Marker Processor
Dim marker As ITemplateMarkersProcessor =
workbook.CreateTemplateMarkersProcessor()
Dim conditionalFormats As IConditionalFormats =
marker.CreateConditionalFormats(worksheet("C5"))
'Region "Data Bar"
'Apply markers using Formula
Dim condition As IConditionalFormat = conditionalFormats.AddCondition()
'Set Data bar and icon set for the same cell
'Set the format type
condition.FormatType = ExcelCfType.DataBar
Dim dataBar As IDataBar = condition.DataBar
'Set the constraint
dataBar.MinPoint.Type = ConditionValueType.LowestValue
dataBar.MinPoint.Value = "0"
dataBar.MaxPoint.Type = ConditionValueType.HighestValue
dataBar.MaxPoint.Value = "0"
'Set color for Bar

```

```

dataBar.BarColor = Color.FromArgb(156, 208, 243)
'Hide the value in data bar
dataBar.ShowValue = False
'End Region
'Region "IconSet"
condition = conditionalFormats.AddCondition()
condition.FormatType = ExcelCfType.IconSet
Dim iconSet As IIconSet = condition.IconSet
iconSet.IconSet = ExcelIconSetType.FourRating
iconSet.IconCriteria(0).Type = ConditionValueType.LowestValue
iconSet.IconCriteria(0).Value = "0"
iconSet.IconCriteria(1).Type = ConditionValueType.HighestValue
iconSet.IconCriteria(1).Value = "0"
iconSet.ShowIconOnly = True
'End Region
conditionalFormats = marker.CreateConditionalFormats(worksheet("D5"))
'Region "Color Scale"
condition = conditionalFormats.AddCondition()
condition.FormatType = ExcelCfType.ColorScale
Dim colorScale As IColorScale = condition.ColorScale
'Sets 3 - color scale
colorScale.SetConditionCount(3)
colorScale.Criteria(0).FormatColorRGB = Color.FromArgb(230, 197, 218)
colorScale.Criteria(0).Type = ConditionValueType.LowestValue
colorScale.Criteria(0).Value = "0"
colorScale.Criteria(1).FormatColorRGB = Color.FromArgb(244, 210, 178)
colorScale.Criteria(1).Type = ConditionValueType.Percentile
colorScale.Criteria(1).Value = "50"
colorScale.Criteria(2).FormatColorRGB = Color.FromArgb(245, 247, 171)
colorScale.Criteria(2).Type = ConditionValueType.HighestValue
colorScale.Criteria(2).Value = "0"
'End Region
conditionalFormats = marker.CreateConditionalFormats(worksheet("E5"))
'Region "IconSet"
condition = conditionalFormats.AddCondition()
condition.FormatType = ExcelCfType.IconSet
iconSet = condition.IconSet
iconSet.IconSet = ExcelIconSetType.ThreeSymbols
iconSet.IconCriteria(0).Type = ConditionValueType.LowestValue
iconSet.IconCriteria(0).Value = "0"
iconSet.IconCriteria(1).Type = ConditionValueType.HighestValue
iconSet.IconCriteria(1).Value = "0"
iconSet.ShowIconOnly = False
'End Region
'Add collection to the marker variables where the name should match with
input template
marker.AddVariable("Reports", GetSalesReports())
'Process the markers in the template
marker.ApplyMarkers()
'End Region
'Region "Save the Workbook"
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("TemplateMarkerCF.xlsx")
'End Region
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("TemplateMarker.TemplateMarker.xlsx");
    IWorkbook workbook = await
        excelEngine.Excel.Workbooks.OpenAsync(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    #endregion
    #region Create Template Marker
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
        workbook.CreateTemplateMarkersProcessor();
    IConditionalFormats conditionalFormats =
        marker.CreateConditionalFormats(worksheet["C5"]);
    #region Data Bar
    //Apply markers using Formula
    IConditionalFormat condition = conditionalFormats.AddCondition();
    //Set Data bar and icon set for the same cell
    //Set the format type
    condition.FormatType = ExcelCFTType.DataBar;
    IDataBar dataBar = condition.DataBar;
    //Set the constraint
    dataBar.MinPoint.Type = ConditionValueType.LowestValue;
    dataBar.MinPoint.Value = "0";
    dataBar.MaxPoint.Type = ConditionValueType.HighestValue;
    dataBar.MaxPoint.Value = "0";
    //Set color for Bar
    dataBar.BarColor = Color.FromArgb(156, 208, 243, 255);
    //Hide the value in data bar
    dataBar.ShowValue = false;
    #endregion
    #region IconSet
    condition = conditionalFormats.AddCondition();
    condition.FormatType = ExcelCFTType.IconSet;
    IIconSet iconSet = condition.IconSet;
    iconSet.IconSet = ExcelIconSetType.FourRating;
    iconSet.IconCriteria[0].Type = ConditionValueType.LowestValue;
    iconSet.IconCriteria[0].Value = "0";
    iconSet.IconCriteria[1].Type = ConditionValueType.HighestValue;
    iconSet.IconCriteria[1].Value = "0";
    iconSet.ShowIconOnly = true;
    #endregion
    conditionalFormats = marker.CreateConditionalFormats(worksheet["D5"]);
    #region Color Scale
    condition = conditionalFormats.AddCondition();
    condition.FormatType = ExcelCFTType.ColorScale;
    IColorScale colorScale = condition.ColorScale;
    //Sets 3 - color scale
    colorScale.SetConditionCount(3);
    colorScale.Criteria[0].FormatColorRGB = Color.FromArgb(230, 197, 218, 255);
    colorScale.Criteria[0].Type = ConditionValueType.LowestValue;
    colorScale.Criteria[0].Value = "0";

```

```

colorScale.Criteria[1].FormatColorRGB = Color.FromArgb(244, 210, 178, 255);
colorScale.Criteria[1].Type = ConditionValueType.Percentile;
colorScale.Criteria[1].Value = "50";
colorScale.Criteria[2].FormatColorRGB = Color.FromArgb(245, 247, 171, 255);
colorScale.Criteria[2].Type = ConditionValueType.HighestValue;
colorScale.Criteria[2].Value = "0";
#endregion
conditionalFormats = marker.CreateConditionalFormats(worksheet["E5"]);
#region IconSet
condition = conditionalFormats.AddCondition();
condition.FormatType = ExcelCFTType.IconSet;
iconSet = condition.IconSet;
iconSet.IconSet = ExcelIconSetType.ThreeSymbols;
iconSet.IconCriteria[0].Type = ConditionValueType.LowestValue;
iconSet.IconCriteria[0].Value = "0";
iconSet.IconCriteria[1].Type = ConditionValueType.HighestValue;
iconSet.IconCriteria[1].Value = "0";
iconSet.ShowIconOnly = false;
#endregion
//Add collection to the marker variables where the name should match with
input template
marker.AddVariable("Reports", GetSalesReports());
//Process the markers in the template
marker.ApplyMarkers();
#endregion
#region Save the Workbook
workbook.Version = ExcelVersion.Excel2013;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "TemplateMarkerCF";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
#endregion
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
#region Initialize Workbook
FileStream fileStream = new FileStream("TemplateMarker.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = excelEngine.Excel.Workbooks.Open(fileStream);
IWorksheet worksheet = workbook.Worksheets[0];
#endregion
#region Create Template Marker
//Create Template Marker Processor
ITemplateMarkersProcessor marker =
workbook.CreateTemplateMarkersProcessor();
IConditionalFormats conditionalFormats =
marker.CreateConditionalFormats(worksheet["C5"]);

```

```

#endregion
#region Data Bar
//Apply markers using Formula
IConditionalFormat condition = conditionalFormats.AddCondition();
//Set Data bar and icon set for the same cell
//Set the format type
condition.FormatType = ExcelCfType.DataBar;
IDataBar dataBar = condition.DataBar;
//Set the constraint
dataBar.MinPoint.Type = ConditionValueType.LowestValue;
dataBar.MinPoint.Value = "0";
dataBar.MaxPoint.Type = ConditionValueType.HighestValue;
dataBar.MaxPoint.Value = "0";
//Set color for Bar
dataBar.BarColor = Color.FromArgb(156, 208, 243);
//Hide the value in data bar
dataBar.ShowValue = false;
#endregion
#region IconSet
condition = conditionalFormats.AddCondition();
condition.FormatType = ExcelCfType.IconSet;
IIconSet iconSet = condition.IconSet;
iconSet.IconSet = ExcelIconSetType.FourRating;
iconSet.IconCriteria[0].Type = ConditionValueType.LowestValue;
iconSet.IconCriteria[0].Value = "0";
iconSet.IconCriteria[1].Type = ConditionValueType.HighestValue;
iconSet.IconCriteria[1].Value = "0";
iconSet.ShowIconOnly = true;
#endregion
conditionalFormats = marker.CreateConditionalFormats(worksheet["D5"]);
#region Color Scale
condition = conditionalFormats.AddCondition();
condition.FormatType = ExcelCfType.ColorScale;
IColorScale colorScale = condition.ColorScale;
//Sets 3 - color scale
colorScale.SetConditionCount(3);
colorScale.Criteria[0].FormatColorRGB = Color.FromArgb(230, 197, 218);
colorScale.Criteria[0].Type = ConditionValueType.LowestValue;
colorScale.Criteria[0].Value = "0";
colorScale.Criteria[1].FormatColorRGB = Color.FromArgb(244, 210, 178);
colorScale.Criteria[1].Type = ConditionValueType.Percentile;
colorScale.Criteria[1].Value = "50";
colorScale.Criteria[2].FormatColorRGB = Color.FromArgb(245, 247, 171);
colorScale.Criteria[2].Type = ConditionValueType.HighestValue;
colorScale.Criteria[2].Value = "0";
#endregion
conditionalFormats = marker.CreateConditionalFormats(worksheet["E5"]);
#region IconSet
condition = conditionalFormats.AddCondition();
condition.FormatType = ExcelCfType.IconSet;
iconSet = condition.IconSet;
iconSet.IconSet = ExcelIconSetType.ThreeSymbols;
iconSet.IconCriteria[0].Type = ConditionValueType.LowestValue;
iconSet.IconCriteria[0].Value = "0";
iconSet.IconCriteria[1].Type = ConditionValueType.HighestValue;
iconSet.IconCriteria[1].Value = "0";
iconSet.ShowIconOnly = false;

```

```

#endregion
//Add collection to the marker variables where the name should match with
input template
marker.AddVariable("Reports", GetSalesReports());
//Process the markers in the template
marker.ApplyMarkers();
//Saving the workbook as stream
workbook.Version = ExcelVersion.Excel2013;
FileStream stream = new FileStream("TemplateMarkerCF.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    #region Initialize Workbook
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("TemplateMarker.xlsx");
    IWorkbook workbook = excelEngine.Excel.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    #endregion
    #region Create Template Marker
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
    workbook.CreateTemplateMarkersProcessor();
    IConditionalFormats conditionalFormats =
    marker.CreateConditionalFormats(worksheet["C5"]);
    #endregion
    #region Data Bar
    //Apply markers using Formula
    IConditionalFormat condition = conditionalFormats.AddCondition();
    //Set Data bar and icon set for the same cell
    //Set the format type
    condition.FormatType = ExcelCFTType.DataBar;
    IDataBar dataBar = condition.DataBar;
    //Set the constraint
    dataBar.MinPoint.Type = ConditionValueType.LowestValue;
    dataBar.MinPoint.Value = "0";
    dataBar.MaxPoint.Type = ConditionValueType.HighestValue;
    dataBar.MaxPoint.Value = "0";
    //Hide the value in data bar
    dataBar.ShowValue = false;
    #endregion
    #region IconSet
    condition = conditionalFormats.AddCondition();
    condition.FormatType = ExcelCFTType.IconSet;
    IIconSet iconSet = condition.IconSet;
    iconSet.IconSet = ExcelIconSetType.FourRating;
    iconSet.IconCriteria[0].Type = ConditionValueType.LowestValue;
    iconSet.IconCriteria[0].Value = "0";

```

```

iconSet.IconCriteria[1].Type = ConditionValueType.HighestValue;
iconSet.IconCriteria[1].Value = "0";
iconSet.ShowIconOnly = true;
#endregion
conditionalFormats = marker.CreateConditionalFormats(worksheet["D5"]);
#region Color Scale
condition = conditionalFormats.AddCondition();
condition.FormatType = ExcelCFTType.ColorScale;
IColorScale colorScale = condition.ColorScale;
//Sets 3 - color scale
colorScale.SetConditionCount(3);
colorScale.Criteria[0].FormatColorRGB =
Syncfusion.Drawing.Color.FromArgb(230, 197, 218);
colorScale.Criteria[0].Type = ConditionValueType.LowestValue;
colorScale.Criteria[0].Value = "0";
colorScale.Criteria[1].FormatColorRGB =
Syncfusion.Drawing.Color.FromArgb(244, 210, 178);
colorScale.Criteria[1].Type = ConditionValueType.Percentile;
colorScale.Criteria[1].Value = "50";
colorScale.Criteria[2].FormatColorRGB =
Syncfusion.Drawing.Color.FromArgb(245, 247, 171);
colorScale.Criteria[2].Type = ConditionValueType.HighestValue;
colorScale.Criteria[2].Value = "0";
#endregion
conditionalFormats = marker.CreateConditionalFormats(worksheet["E5"]);
#region IconSet
condition = conditionalFormats.AddCondition();
condition.FormatType = ExcelCFTType.IconSet;
iconSet = condition.IconSet;
iconSet.IconSet = ExcelIconSetType.ThreeSymbols;
iconSet.IconCriteria[0].Type = ConditionValueType.LowestValue;
iconSet.IconCriteria[0].Value = "0";
iconSet.IconCriteria[1].Type = ConditionValueType.HighestValue;
iconSet.IconCriteria[1].Value = "0";
iconSet.ShowIconOnly = false;
#endregion
//Add collection to the marker variables where the name should match with
input template
marker.AddVariable("Reports", GetSalesReports());
//Process the markers in the template
marker.ApplyMarkers();
//Saving the workbook as stream
workbook.Version = ExcelVersion.Excel2013;
MemoryStream outputStream = new MemoryStream();
workbook.SaveAs(outputStream);
//Save the stream as Excel document and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
await
DependencyService.Get<ISaveWindowsPhone>().SaveAndView("TemplateMarker.xlsx"
, "application/msexcel", outputStream);
else
DependencyService.Get<ISave>().SaveAndView("TemplateMarker.xlsx",
"application/msexcel", outputStream);

```

```
//Dispose the input and output stream instances
inputStream.Dispose();
outputStream.Dispose();
}
```

The following code snippet provides supporting method and class for the previous code.

GetSalesReports Method:

C#

```
public IList<Sales> GetSalesReports()
{
    IList<Sales> sales = new List<Sales>();
    sales.Add(new Sales("Andy Bernard", 45000, 58000, 29));
    sales.Add(new Sales("Jim Halpert", 34000, 65000, 91));
    sales.Add(new Sales("Karen Fillippelli", 75000, 64000, -15));
    sales.Add(new Sales("Phyllis Lapin", 56500, 33600, -40));
    sales.Add(new Sales("Stanley Hudson", 46500, 52000, 12));
    return sales;
}

public class Sales
{
    public string SalesPerson { get; set; }
    public int SalesJanJun { get; set; }
    public int SalesJulDec { get; set; }
    public int Change { get; set; }
    public Sales(string name, int salesJanJun, int salesJulDec, int change)
    {
        SalesPerson = name;
        SalesJanJun = salesJanJun;
        SalesJulDec = salesJulDec;
        Change = change;
    }
}
```

VB.NET

```
Public Function GetSalesReports() As IList(Of Sales)
Dim sales As IList(Of Sales) = New List(Of Sales)()
sales.Add(New Sales("Andy Bernard", 45000, 58000, 29))
sales.Add(New Sales("Jim Halpert", 34000, 65000, 91))
sales.Add(New Sales("Karen Fillippelli", 75000, 64000, -15))
sales.Add(New Sales("Phyllis Lapin", 56500, 33600, -40))
sales.Add(New Sales("Stanley Hudson", 46500, 52000, 12))
Return sales
End Function

Public Class Sales
Public Property SalesPerson As String
Public Property SalesJanJun As Integer
Public Property SalesJulDec As Integer
Public Property Change As Integer
Public Sub New(ByVal name As String, ByVal salesJanJun As Integer, ByVal
salesJulDec As Integer, ByVal change As Integer)
SalesPerson = name
salesJanJun = salesJanJun
```



```

salesJulDec = salesJulDec
change = change
End Sub
End Class

```

UWP

```

public IList<Sales> GetSalesReports()
{
    IList<Sales> sales = new List<Sales>();
    sales.Add(new Sales("Andy Bernard", 45000, 58000, 29));
    sales.Add(new Sales("Jim Halpert", 34000, 65000, 91));
    sales.Add(new Sales("Karen Fillippelli", 75000, 64000, -15));
    sales.Add(new Sales("Phyllis Lapin", 56500, 33600, -40));
    sales.Add(new Sales("Stanley Hudson", 46500, 52000, 12));
    return sales;
}

public class Sales
{
    public string SalesPerson { get; set; }
    public int SalesJanJun { get; set; }
    public int SalesJulDec { get; set; }
    public int Change { get; set; }
    public Sales(string name, int salesJanJun, int salesJulDec, int change)
    {
        SalesPerson = name;
        SalesJanJun = salesJanJun;
        SalesJulDec = salesJulDec;
        Change = change;
    }
}

```

ASP.NET CORE

```

public IList<Sales> GetSalesReports()
{
    IList<Sales> sales = new List<Sales>();
    sales.Add(new Sales("Andy Bernard", 45000, 58000, 29));
    sales.Add(new Sales("Jim Halpert", 34000, 65000, 91));
    sales.Add(new Sales("Karen Fillippelli", 75000, 64000, -15));
    sales.Add(new Sales("Phyllis Lapin", 56500, 33600, -40));
    sales.Add(new Sales("Stanley Hudson", 46500, 52000, 12));
    return sales;
}

public class Sales
{
    public string SalesPerson { get; set; }
    public int SalesJanJun { get; set; }
    public int SalesJulDec { get; set; }
    public int Change { get; set; }
    public Sales(string name, int salesJanJun, int salesJulDec, int change)
    {
        SalesPerson = name;
        SalesJanJun = salesJanJun;
        SalesJulDec = salesJulDec;
        Change = change;
    }
}

```

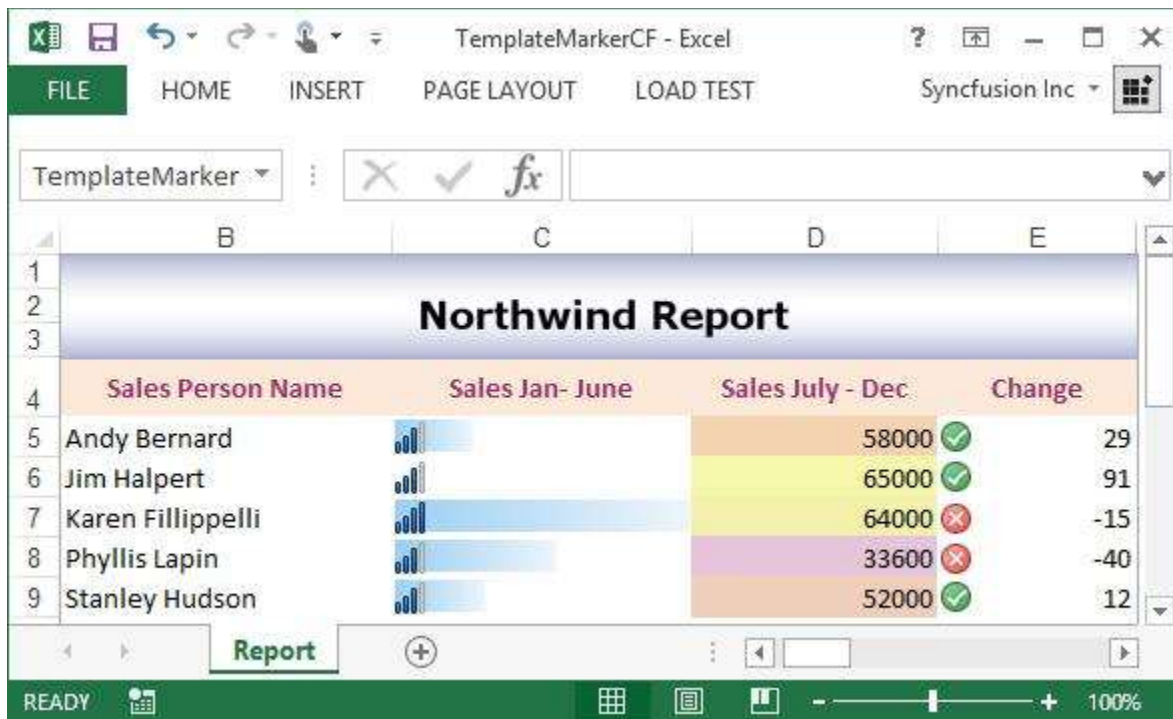
```
}
}
```

XAMARIN

```
public IList<Sales> GetSalesReports()
{
    IList<Sales> sales = new List<Sales>();
    sales.Add(new Sales("Andy Bernard", 45000, 58000, 29));
    sales.Add(new Sales("Jim Halpert", 34000, 65000, 91));
    sales.Add(new Sales("Karen Fillippelli", 75000, 64000, -15));
    sales.Add(new Sales("Phyllis Lapin", 56500, 33600, -40));
    sales.Add(new Sales("Stanley Hudson", 46500, 52000, 12));
    return sales;
}

public class Sales
{
    public string SalesPerson { get; set; }
    public int SalesJanJun { get; set; }
    public int SalesJulDec { get; set; }
    public int Change { get; set; }
    public Sales(string name, int salesJanJun, int salesJulDec, int change)
    {
        SalesPerson = name;
        SalesJanJun = salesJanJun;
        SalesJulDec = salesJulDec;
        Change = change;
    }
}
```

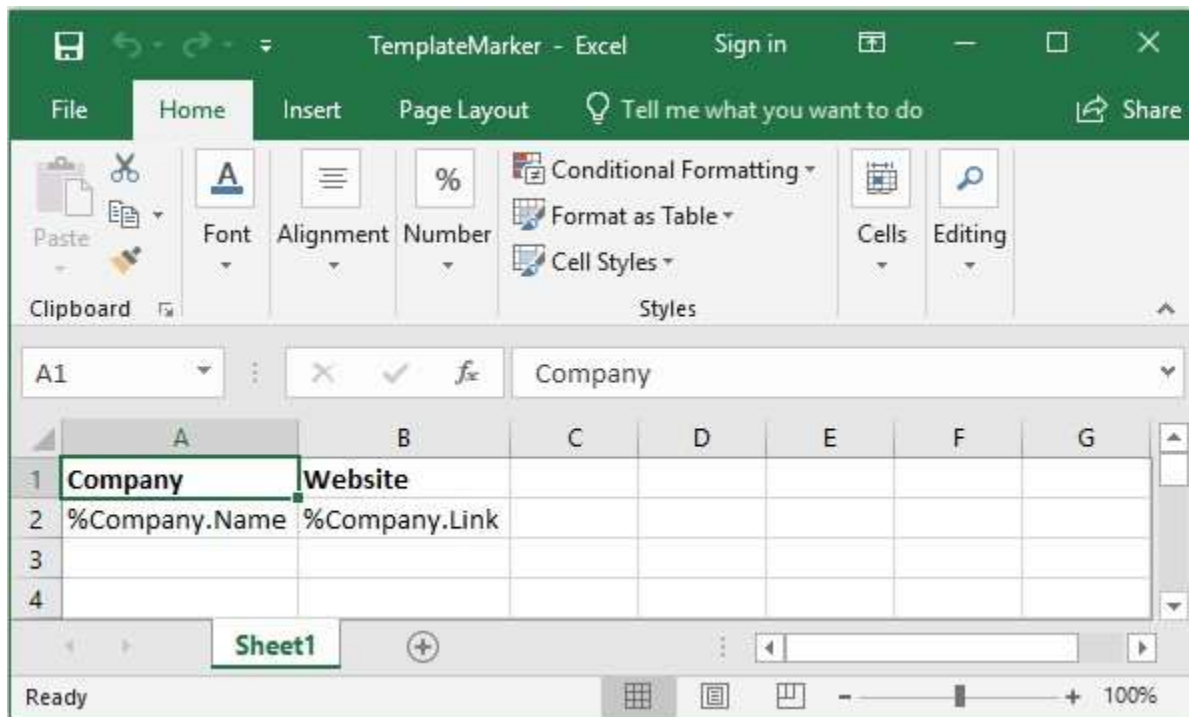
The following screenshot represents generated Excel file in which the conditional format is applied.



Template marker with Hyperlink

You can add hyperlink to the template marker range.

The following screenshot represents the input template, which has a template marker.



The following code snippet illustrates how to detect data type and apply number format with template marker.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("TemplateMarker.xlsx")
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
    workbook.CreateTemplateMarkersProcessor();
    //Add collection to the marker variables where the name should match with
    input template
    marker.AddVariable("Company", GetCompanyDetails());
    //Process the markers in the template
    marker.ApplyMarkers();
    workbook.SaveAs("TemplateMarkerHyperlink.xlsx");
}
```

VB.NET

```
using excelEngine As ExcelEngine = new ExcelEngine()
Dim workbook As IWorkbook =
excelEngine.Excel.Workbooks.Open("TemplateMarker.xlsx")
'Create Template Marker Processor
```

```

Dim marker As ITemplateMarkersProcessor =
workbook.CreateTemplateMarkersProcessor()
'Add collection to the marker variables where the name should match with
input template
marker.AddVariable("Company", GetCompanyDetails());
'Process the markers and detect the number format along with the data type
in the template
marker.ApplyMarkers()
workbook.SaveAs("TemplateMarkerHyperlink.xlsx");
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile openFile = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(openFile);
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
workbook.CreateTemplateMarkersProcessor();
    //Add collection to the marker variables where the name should match with
input template
    marker.AddVariable("Company", GetCompanyDetails());
    //Process the markers in the template
    marker.ApplyMarkers();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "TemplateMarkerHyperlink";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

//Binding data from data table is supported only from ASP.NET Core 2.0
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("TemplateMarker.xlsx", FileMode.Open,
FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
workbook.CreateTemplateMarkersProcessor();

```

```
//Add collection to the marker variables where the name should match with
input template
marker.AddVariable("Company", GetCompanyDetails());
//Process the markers in the template
marker.ApplyMarkers();
//Saving the workbook as stream
FileStream stream = new FileStream("TemplateMarkerHyperlink.xlsx",
    FileMode.Create, FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.TemplateMarker.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    //Create Template Marker Processor
    ITemplateMarkersProcessor marker =
        workbook.CreateTemplateMarkersProcessor();
    //Add collection to the marker variables where the name should match with
    input template
    marker.AddVariable("Company", GetCompanyDetails());
    //Process the markers in the template
    marker.ApplyMarkers();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("TemplateMarkerHyperlink.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TemplateMarkerHyperlink.xlsx", "application/msexcel", stream);
    }
}
```

The following code snippet provides supporting methods and classes for the previous code.

C#

```

//Gets a list of company details
private List<Company> GetCompanyDetails()
{
    List<Company> companyList = new List<Company>();
    Company company = new Company();
    company.Name = "Syncfusion";
    Hyperlink link = new Hyperlink("https://www.syncfusion.com", "", "",
    "Syncfusion", ExcelHyperLinkType.Url, null);
    company.Link = link;
    companyList.Add(company);
    company = new Company();
    company.Name = "Microsoft";
    link = new Hyperlink("https://www.microsoft.com", "", "", "Microsoft",
    ExcelHyperLinkType.Url, null);
    company.Link = link;
    companyList.Add(company);
    company = new Company();
    company.Name = "Google";
    link = new Hyperlink("https://www.google.com", "", "", "Google",
    ExcelHyperLinkType.Url, null);
    company.Link = link;
    companyList.Add(company);
    return companyList;
}

public class Hyperlink : IHyperLink
{
    public IApplication Application { get; }
    public object Parent { get; }
    public string Address { get; set; }
    public string Name { get; }
    public IRange Range { get; }
    public string ScreenTip { get; set; }
    public string SubAddress { get; set; }
    public string TextToDisplay { get; set; }
    public ExcelHyperLinkType Type { get; set; }
    public IShape Shape { get; }
    public ExcelHyperlinkAttachedType AttachedType { get; }
    public byte[] Image { get; set; }
    public Hyperlink(string address, string subAddress, string screenTip, string
    textToDisplay, ExcelHyperLinkType type, byte[] image)
    {
        Address = address;
        ScreenTip = screenTip;
        SubAddress = subAddress;
        TextToDisplay = textToDisplay;
        Type = type;
        Image = image;
    }
}

public class Company
{
    public string Name { get; set; }
    public Hyperlink Link { get; set; }
}

```

VB.NET

```

'Gets a list of company details
Private Function GetCompanyDetails() As List(Of Company)
Dim companyList As List(Of Company) = New List(Of Company)()
Dim company As Company = New Company()
company.Name = "Syncfusion"
Dim link As Hyperlink = New Hyperlink("https://www.syncfusion.com", "", "",
"Syncfusion", ExcelHyperLinkType.Url, Nothing)
company.Link = link
companyList.Add(company)
company = New Company()
company.Name = "Microsoft"
link = New Hyperlink("https://www.microsoft.com", "", "", "Microsoft",
ExcelHyperLinkType.Url, Nothing)
company.Link = link
companyList.Add(company)
company = New Company()
company.Name = "Google"
link = New Hyperlink("https://www.google.com", "", "", "Google",
ExcelHyperLinkType.Url, Nothing)
company.Link = link
companyList.Add(company)
Return companyList
End Function
Public Class Hyperlink
Inherits IHyperLink
Public ReadOnly Property Application As IApplication
Public ReadOnly Property Parent As Object
Public Property Address As String
Public ReadOnly Property Name As String
Public ReadOnly Property Range As IRange
Public Property ScreenTip As String
Public Property SubAddress As String
Public Property TextToDisplay As String
Public Property Type As ExcelHyperLinkType
Public ReadOnly Property Shape As IShape
Public ReadOnly Property AttachedType As ExcelHyperlinkAttachedType
Public Property Image As Byte()
Public Sub New(ByVal address As String, ByVal subAddress As String, ByVal
screenTip As String, ByVal textToDisplay As String, ByVal type As
ExcelHyperLinkType, ByVal image As Byte())
Address = address
ScreenTip = screenTip
SubAddress = subAddress
TextToDisplay = textToDisplay
Type = type
Image = image
End Sub
End Class
Public Class Company
Public Property Name As String
Public Property Link As Hyperlink
End Class

```

UWP

```

//Gets a list of company details

```

```

private List<Company> GetCompanyDetails()
{
    List<Company> companyList = new List<Company>();
    Company company = new Company();
    company.Name = "Syncfusion";
    Hyperlink link = new Hyperlink("https://www.syncfusion.com", "", "",
    "Syncfusion", ExcelHyperLinkType.Url, null);
    company.Link = link;
    companyList.Add(company);
    company = new Company();
    company.Name = "Microsoft";
    link = new Hyperlink("https://www.microsoft.com", "", "", "Microsoft",
    ExcelHyperLinkType.Url, null);
    company.Link = link;
    companyList.Add(company);
    company = new Company();
    company.Name = "Google";
    link = new Hyperlink("https://www.google.com", "", "", "Google",
    ExcelHyperLinkType.Url, null);
    company.Link = link;
    companyList.Add(company);
    return companyList;
}

public class Hyperlink : IHyperLink
{
    public IApplication Application { get; }
    public object Parent { get; }
    public string Address { get; set; }
    public string Name { get; }
    public IRange Range { get; }
    public string ScreenTip { get; set; }
    public string SubAddress { get; set; }
    public string TextToDisplay { get; set; }
    public ExcelHyperLinkType Type { get; set; }
    public IShape Shape { get; }
    public ExcelHyperlinkAttachedType AttachedType { get; }
    public byte[] Image { get; set; }
    public Hyperlink(string address, string subAddress, string screenTip, string
    textToDisplay, ExcelHyperLinkType type, byte[] image)
    {
        Address = address;
        ScreenTip = screenTip;
        SubAddress = subAddress;
        TextToDisplay = textToDisplay;
        Type = type;
        Image = image;
    }
}

public class Company
{
    public string Name { get; set; }
    public Hyperlink Link { get; set; }
}

```

ASP.NET CORE


```
//Gets a list of company details
private List<Company> GetCompanyDetails()
{
    List<Company> companyList = new List<Company>();
    Company company = new Company();
    company.Name = "Syncfusion";
    Hyperlink link = new Hyperlink("https://www.syncfusion.com", "", "",
    "Syncfusion", ExcelHyperLinkType.Url, null);
    company.Link = link;
    companyList.Add(company);
    company = new Company();
    company.Name = "Microsoft";
    link = new Hyperlink("https://www.microsoft.com", "", "", "Microsoft",
    ExcelHyperLinkType.Url, null);
    company.Link = link;
    companyList.Add(company);
    company = new Company();
    company.Name = "Google";
    link = new Hyperlink("https://www.google.com", "", "", "Google",
    ExcelHyperLinkType.Url, null);
    company.Link = link;
    companyList.Add(company);
    return companyList;
}

public class Hyperlink : IHyperLink
{
    public IApplication Application { get; }
    public object Parent { get; }
    public string Address { get; set; }
    public string Name { get; }
    public IRange Range { get; }
    public string ScreenTip { get; set; }
    public string SubAddress { get; set; }
    public string TextToDisplay { get; set; }
    public ExcelHyperLinkType Type { get; set; }
    public IShape Shape { get; }
    public ExcelHyperlinkAttachedType AttachedType { get; }
    public byte[] Image { get; set; }
    public Hyperlink(string address, string subAddress, string screenTip, string
    textToDisplay, ExcelHyperLinkType type, byte[] image)
    {
        Address = address;
        ScreenTip = screenTip;
        SubAddress = subAddress;
        TextToDisplay = textToDisplay;
        Type = type;
        Image = image;
    }
}

public class Company
{
    public string Name { get; set; }
    public Hyperlink Link { get; set; }
}
```

XAMARIN

```

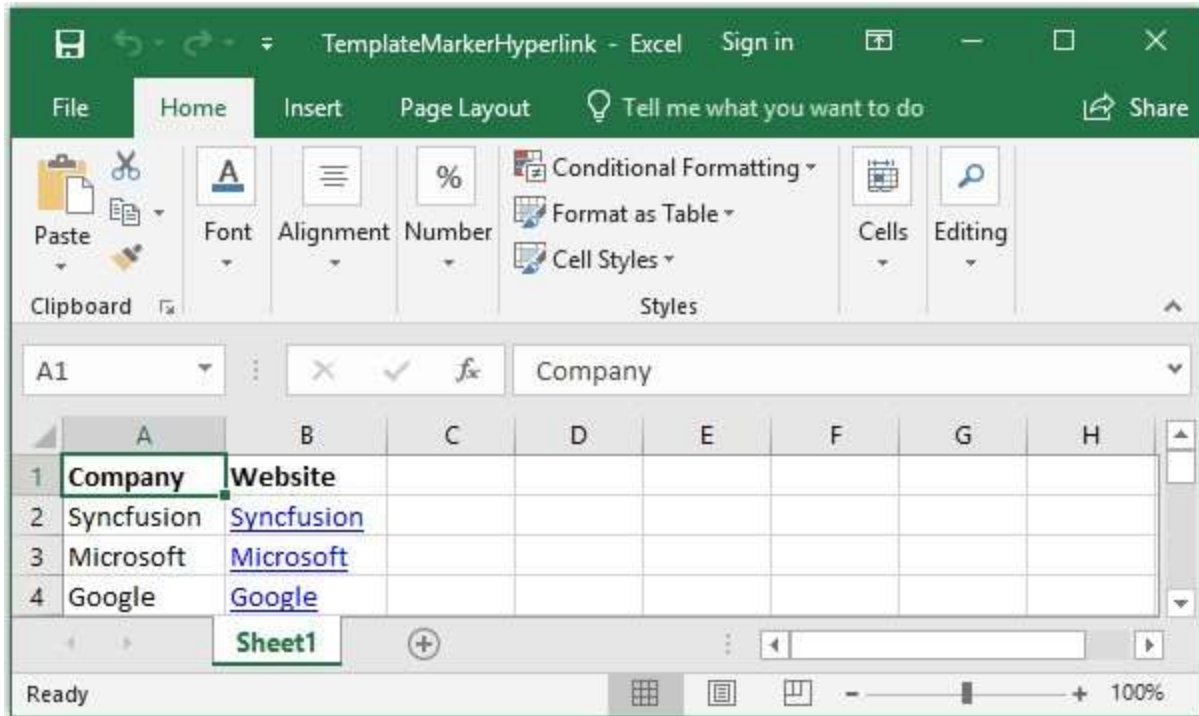
//Gets a list of company details
private List<Company> GetCompanyDetails()
{
    List<Company> companyList = new List<Company>();
    Company company = new Company();
    company.Name = "Syncfusion";
    Hyperlink link = new Hyperlink("https://www.syncfusion.com", "", "",
    "Syncfusion", ExcelHyperLinkType.Url, null);
    company.Link = link;
    companyList.Add(company);
    company = new Company();
    company.Name = "Microsoft";
    link = new Hyperlink("https://www.microsoft.com", "", "", "Microsoft",
    ExcelHyperLinkType.Url, null);
    company.Link = link;
    companyList.Add(company);
    company = new Company();
    company.Name = "Google";
    link = new Hyperlink("https://www.google.com", "", "", "Google",
    ExcelHyperLinkType.Url, null);
    company.Link = link;
    companyList.Add(company);
    return companyList;
}

public class Hyperlink : IHyperLink
{
    public IApplication Application { get; }
    public object Parent { get; }
    public string Address { get; set; }
    public string Name { get; }
    public IRange Range { get; }
    public string ScreenTip { get; set; }
    public string SubAddress { get; set; }
    public string TextToDisplay { get; set; }
    public ExcelHyperLinkType Type { get; set; }
    public IShape Shape { get; }
    public ExcelHyperlinkAttachedType AttachedType { get; }
    public byte[] Image { get; set; }
    public Hyperlink(string address, string subAddress, string screenTip, string
    textToDisplay, ExcelHyperLinkType type, byte[] image)
    {
        Address = address;
        ScreenTip = screenTip;
        SubAddress = subAddress;
        TextToDisplay = textToDisplay;
        Type = type;
        Image = image;
    }
}

public class Company
{
    public string Name { get; set; }
    public Hyperlink Link { get; set; }
}

```

The following screenshot represents generated Excel file in which the hyperlink is added.



Working with Tables

Creating a table

XlsIO supports reading and writing the table which helps to organize and analyze the related data.

- **IListObjects** represents a collection of tables in the worksheet.
- **IListObject** represent a table in the worksheet.

You can also create a calculated column in the table. For more details, refer [here](#).

Note: In XlsIO, tables are supported only for Excel 2007 and later formats (*.xlsx files).

The following code sample explains the creation of a simple table by the range of data from an existing worksheet.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create table with the data in given range
    IListObject table = worksheet.ListObjects.Create("Table1",
        worksheet["A1:C8"]);
    string fileName = "Output.xlsx";
    workbook.SaveAs(fileName);
}
```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Create table with the data in given range
Dim table As IListObject = worksheet.ListObjects.Create("Table1",
worksheet("A1:C8"))
Dim fileName As String = "Output.xlsx"
workbook.SaveAs(fileName)
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from an embedded resource collection
Stream inputStream =
assembly.GetManifestResourceStream("Table.Sample.xlsx");
IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Create table with the data in given range
IListObject table = worksheet.ListObjects.Create("Table1",
worksheet["A1:C8"]);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Create table with the data in given range

```

```

IListObject table = worksheet.ListObjects.Create("Table1",
worksheet["A1:C8"]);
string fileName = "Output.xlsx";
//Saving the workbook as stream
FileStream stream = new FileStream(fileName, FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("Table.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create table with the data in given range
    IListObject table = worksheet.ListObjects.Create("Table1",
    worksheet["A1:C8"]);
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
    string fileName = "Output.xlsx";
    outputStream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
        "application/msexcel", outputStream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
        "application/msexcel", outputStream);
    }
}

```

Accessing a table

The existing tables in the worksheet can be accessed, as follows.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{

```

```

IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Accessing first table in the sheet
IListObject table = worksheet.ListObjects[0];
//Modifying table name
table.Name = "SalesTable";
string fileName = "Output.xlsx";
workbook.SaveAs(fileName);
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Accessing first table in the sheet
Dim table As IListObject = worksheet.ListObjects(0)
'Modifying table name
table.Name = "SalesTable"
Dim fileName As String = "Output.xlsx"
workbook.SaveAs(fileName)
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from an embedded resource collection
Stream inputStream =
assembly.GetManifestResourceStream("Table.Sample.xlsx");
IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Accessing first table in the sheet
IListObject table = worksheet.ListObjects[0];
//Modifying table name
table.Name = "SalesTable";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from the FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
}

```

```
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream,
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Accessing first table in the sheet
    IListObject table = worksheet.ListObjects[0];
    //Modifying table name
    table.Name = "SalesTable";
    string fileName = "Output.xlsx";
    //Saving the workbook as stream
    FileStream stream = new FileStream(fileName, FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("Table.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Accessing first table in the sheet
    IListObject table = worksheet.ListObjects[0];
    //Modifying table name
    table.Name = "SalesTable";
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
    string fileName = "Output.xlsx";
    outputStream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName, "application/msexcel", outputStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName, "application/msexcel", outputStream);
}
}
```

Formatting a table

You can apply built-in styles to the table using XlsIO. You can also customize the table with other table style options such as Header/total row, first/last column, and banded rows to make a table easier to read.

The following code snippet illustrates how to apply built-in table style.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Creating a table
IListObject table = worksheet.ListObjects.Create("Table1",
worksheet["A1:C8"]);
//Formatting table with a built-in style
table.BuiltInTableStyle = TableBuiltInStyles.TableStyleMedium9;
string fileName = "Output.xlsx";
workbook.SaveAs(fileName);
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Creating a table
Dim table As IListObject = worksheet.ListObjects.Create("Table1",
worksheet("A1:C8"))
'Formatting table with a built-in style
table.BuiltInTableStyle = TableBuiltInStyles.TableStyleMedium9
Dim fileName As String = "Output.xlsx"
workbook.SaveAs(fileName)
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
```



```

{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("Table.Sample.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating a table
    IListObject table = worksheet.ListObjects.Create("Table1",
        worksheet["A1:C8"]);
    //Formatting table with a built-in style
    table.BuiltInTableStyle = TableBuiltInStyles.TableStyleMedium9;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream,
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating a table
    IListObject table = worksheet.ListObjects.Create("Table1",
        worksheet["A1:C8"]);
    //Formatting table with a built-in style
    table.BuiltInTableStyle = TableBuiltInStyles.TableStyleMedium9;
    string fileName = "Output.xlsx";
    //Saving the workbook as stream
    FileStream stream = new FileStream(fileName, FileMode.Create,
        FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{

```

```

IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from embedded resource collection
Stream inputStream =
assembly.GetManifestResourceStream("Table.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Creating a table
IListObject table = worksheet.ListObjects.Create("Table1",
worksheet["A1:C8"]);
//Formatting table with a built-in style
table.BuiltInTableStyle = TableBuiltInStyles.TableStyleMedium9;
//Saving the workbook as stream
MemoryStream outputStream = new MemoryStream();
workbook.SaveAs(outputStream);
string fileName = "Output.xlsx";
outputStream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies among Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
"application/msexcel", outputStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
"application/msexcel", outputStream);
}
}

```

Apply Custom Table Style

You can apply custom table style to the table using XlsIO. You can create custom table style in which you can specified border, font, back ground color and format. You can also customized table in with other table style options such as Header/total row, first/last column, banded rows q to make a table easier to read.

The below code example shows how to apply custom table style in XlsIO.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Create data
worksheet[1, 1].Text = "Products";
worksheet[1, 2].Text = "Qtr1";
}

```

```

worksheet[1, 3].Text = "Qtr2";
worksheet[1, 4].Text = "Qtr3";
worksheet[1, 5].Text = "Qtr4";
worksheet[2, 1].Text = "Alfreds Futterkiste";
worksheet[2, 2].Number = 744.6;
worksheet[2, 3].Number = 162.56;
worksheet[2, 4].Number = 5079.6;
worksheet[2, 5].Number = 1249.2;
worksheet[3, 1].Text = "Antonio Moreno";
worksheet[3, 2].Number = 5079.6;
worksheet[3, 3].Number = 1249.2;
worksheet[3, 4].Number = 943.89;
worksheet[3, 5].Number = 349.6;
worksheet[4, 1].Text = "Around the Horn";
worksheet[4, 2].Number = 1267.5;
worksheet[4, 3].Number = 1062.5;
worksheet[4, 4].Number = 744.6;
worksheet[4, 5].Number = 162.56;
worksheet[5, 1].Text = "Bon app";
worksheet[5, 2].Number = 1418;
worksheet[5, 3].Number = 756;
worksheet[5, 4].Number = 1267.5;
worksheet[5, 5].Number = 1062.5;
worksheet[6, 1].Text = "Eastern Connection";
worksheet[6, 2].Number = 4728;
worksheet[6, 3].Number = 4547.92;
worksheet[6, 4].Number = 1418;
worksheet[6, 5].Number = 756;
worksheet[7, 1].Text = "Ernst Handel";
worksheet[7, 2].Number = 943.89;
worksheet[7, 3].Number = 349.6;
worksheet[7, 4].Number = 4728;
worksheet[7, 5].Number = 4547.92;
//Create style for table number format
IStyle style = workbook.Styles.Add("CurrencyFormat");
style.NumberFormat = "_(($* #,##0.00_);_($* (#,##0.00);_($* \" -
\"??_);_(@_))";
worksheet["B2:E8"].CellStyleName = "CurrencyFormat";
//Create table
IListObject table = worksheet.ListObjects.Create("Table1",
worksheet["A1:E7"]);
//Apply custom table style
ITableStyles tableStyles = workbook.TableStyles;
ITableStyle tableStyle = tableStyles.Add("Table Style 1");
ITableStyleElements tableStyleElements = tableStyle.TableStyleElements;
ITableStyleElement tableStyleElement =
tableStyleElements.Add(ExcelTableStyleElementType.SecondColumnStripe);
tableStyleElement.BackColorRGB = Color.FromArgb(217, 225, 242);
ITableStyleElement tableStyleElement1 =
tableStyleElements.Add(ExcelTableStyleElementType.FirstColumn);
tableStyleElement1.FontColorRGB = Color.FromArgb(128, 128, 128);
ITableStyleElement tableStyleElement2 =
tableStyleElements.Add(ExcelTableStyleElementType.HeaderRow);
tableStyleElement2.FontColor = ExcelKnownColors.White;
tableStyleElement2.BackColorRGB = Color.FromArgb(0, 112, 192);
ITableStyleElement tableStyleElement3 =
tableStyleElements.Add(ExcelTableStyleElementType.TotalRow);

```

```

tableStyleElement3.BackColorRGB = Color.FromArgb(0, 112, 192);
tableStyleElement3.FontColor = ExcelKnownColors.White;
table.TableName = tableStyle.Name;
//Total row
table.ShowTotals = true;
table.ShowFirstColumn = true;
table.ShowTableStyleColumnStripes = true;
table.ShowTableStyleRowStripes = true;
table.Columns[0].TotalsRowLabel = "Total";
table.Columns[1].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[2].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[3].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[4].TotalsCalculation = ExcelTotalsCalculation.Sum;
//Save the workbook
workbook.SaveAs("CustomTableStyle.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Create data
worksheet(1, 1).Text = "Products"
worksheet(1, 2).Text = "Qtr1"
worksheet(1, 3).Text = "Qtr2"
worksheet(1, 4).Text = "Qtr3"
worksheet(1, 5).Text = "Qtr4"
worksheet(2, 1).Text = "Alfreds Futterkiste"
worksheet(2, 2).Number = 744.6
worksheet(2, 3).Number = 162.56
worksheet(2, 4).Number = 5079.6
worksheet(2, 5).Number = 1249.2
worksheet(3, 1).Text = "Antonio Moreno"
worksheet(3, 2).Number = 5079.6
worksheet(3, 3).Number = 1249.2
worksheet(3, 4).Number = 943.89
worksheet(3, 5).Number = 349.6
worksheet(4, 1).Text = "Around the Horn"
worksheet(4, 2).Number = 1267.5
worksheet(4, 3).Number = 1062.5
worksheet(4, 4).Number = 744.6
worksheet(4, 5).Number = 162.56
worksheet(5, 1).Text = "Bon app"
worksheet(5, 2).Number = 1418
worksheet(5, 3).Number = 756
worksheet(5, 4).Number = 1267.5
worksheet(5, 5).Number = 1062.5
worksheet(6, 1).Text = "Eastern Connection"
worksheet(6, 2).Number = 4728
worksheet(6, 3).Number = 4547.92
worksheet(6, 4).Number = 1418
worksheet(6, 5).Number = 756
worksheet(7, 1).Text = "Ernst Handel"

```

```

worksheet(7, 2).Number = 943.89
worksheet(7, 3).Number = 349.6
worksheet(7, 4).Number = 4728
worksheet(7, 5).Number = 4547.92
'Create style for table number format
Dim style As IStyle = workbook.Styles.Add("CurrencyFormat")
style.NumberFormat = "_(($* #,##0.00_);_($* (#,##0.00);_($* "" -
""??_);_(@_)"
worksheet("B2:E8").CellStyleName = "CurrencyFormat"
'Create table
Dim table As IListObject = worksheet.ListObjects.Create("Table1",
worksheet("A1:E7"))
//Apply custom table style
Dim tableStyles As ITableStyles = workbook.TableStyles
Dim tableStyle As ITableStyle = tableStyles.Add("Table Style 1")
Dim tableStyleElements As ITableStyleElements =
tableStyle.TableStyleElements
Dim tableStyleElement As ITableStyleElement =
tableStyleElements.Add(ExcelTableStyleElementType.SecondColumnStripe)
tableStyleElement.BackColorRGB = Color.FromArgb(217, 225, 242)
Dim tableStyleElement1 As ITableStyleElement =
tableStyleElements.Add(ExcelTableStyleElementType.FirstColumn)
tableStyleElement1.FontColorRGB = Color.FromArgb(128, 128, 128)
Dim tableStyleElement2 As ITableStyleElement =
tableStyleElements.Add(ExcelTableStyleElementType.HeaderRow)
tableStyleElement2.FontColor = ExcelKnownColors.White
tableStyleElement2.BackColorRGB = Color.FromArgb(0, 112, 192)
Dim tableStyleElement3 As ITableStyleElement =
tableStyleElements.Add(ExcelTableStyleElementType.TotalRow)
tableStyleElement3.BackColorRGB = Color.FromArgb(0, 112, 192)
tableStyleElement3.FontColor = ExcelKnownColors.White
table.TableName = tableStyle.Name
'Total row
table.ShowTotals = True
table.ShowFirstColumn = True
table.ShowTableStyleColumnStripes = True
table.ShowTableStyleRowStripes = True
table.Columns(0).TotalsRowLabel = "Total"
table.Columns(1).TotalsCalculation = ExcelTotalsCalculation.Sum
table.Columns(2).TotalsCalculation = ExcelTotalsCalculation.Sum
table.Columns(3).TotalsCalculation = ExcelTotalsCalculation.Sum
table.Columns(4).TotalsCalculation = ExcelTotalsCalculation.Sum
'Save the workbook
workbook.SaveAs("CustomTableStyle.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    // Create data
    worksheet[1, 1].Text = "Products";
}

```

```

worksheet[1, 2].Text = "Qtr1";
worksheet[1, 3].Text = "Qtr2";
worksheet[1, 4].Text = "Qtr3";
worksheet[1, 5].Text = "Qtr4";
worksheet[2, 1].Text = "Alfreds Futterkiste";
worksheet[2, 2].Number = 744.6;
worksheet[2, 3].Number = 162.56;
worksheet[2, 4].Number = 5079.6;
worksheet[2, 5].Number = 1249.2;
worksheet[3, 1].Text = "Antonio Moreno";
worksheet[3, 2].Number = 5079.6;
worksheet[3, 3].Number = 1249.2;
worksheet[3, 4].Number = 943.89;
worksheet[3, 5].Number = 349.6;
worksheet[4, 1].Text = "Around the Horn";
worksheet[4, 2].Number = 1267.5;
worksheet[4, 3].Number = 1062.5;
worksheet[4, 4].Number = 744.6;
worksheet[4, 5].Number = 162.56;
worksheet[5, 1].Text = "Bon app";
worksheet[5, 2].Number = 1418;
worksheet[5, 3].Number = 756;
worksheet[5, 4].Number = 1267.5;
worksheet[5, 5].Number = 1062.5;
worksheet[6, 1].Text = "Eastern Connection";
worksheet[6, 2].Number = 4728;
worksheet[6, 3].Number = 4547.92;
worksheet[6, 4].Number = 1418;
worksheet[6, 5].Number = 756;
worksheet[7, 1].Text = "Ernst Handel";
worksheet[7, 2].Number = 943.89;
worksheet[7, 3].Number = 349.6;
worksheet[7, 4].Number = 4728;
worksheet[7, 5].Number = 4547.92;
//Create style for table number format
IStyle style = workbook.Styles.Add("CurrencyFormat");
style.NumberFormat = "_($* #,##0.00_);_($* (#,##0.00);_($* \" -
\"??_);_(@_)";
worksheet["B2:E8"].CellStyleName = "CurrencyFormat";
//Create table
IListObject table = worksheet.ListObjects.Create("Table1",
worksheet["A1:E7"]);
//Apply custom table style
ITableStyles tableStyles = workbook.TableStyles;
ITableStyle tableStyle = tableStyles.Add("Table Style 1");
ITableStyleElements tableStyleElements = tableStyle.TableStyleElements;
ITableStyleElement tableStyleElement =
tableStyleElements.Add(ExcelTableStyleElementType.SecondColumnStripe);
tableStyleElement.BackColorRGB = Color.FromArgb(217, 225, 242);
ITableStyleElement tableStyleElement1 =
tableStyleElements.Add(ExcelTableStyleElementType.FirstColumn);
tableStyleElement1.FontColorRGB = Color.FromArgb(128, 128, 128);
ITableStyleElement tableStyleElement2 =
tableStyleElements.Add(ExcelTableStyleElementType.HeaderRow);
tableStyleElement2.FontColor = ExcelKnownColors.White;
tableStyleElement2.BackColorRGB = Color.FromArgb(0, 112, 192);

```

```

ITableStyleElement tableStyleElement3 =
tableStyleElements.Add(ExcelTableStyleElementType.TotalRow);
tableStyleElement3.BackColorRGB = Color.FromArgb(0, 112, 192);
tableStyleElement3.FontColor = ExcelKnownColors.White;
table.TableStyleName = tableStyle.Name;
//Total row
table.ShowTotals = true;
table.ShowFirstColumn = true;
table.ShowTableStyleColumnStripes = true;
table.ShowTableStyleRowStripes = true;
table.Columns[0].TotalsRowLabel = "Total";
table.Columns[1].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[2].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[3].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[4].TotalsCalculation = ExcelTotalsCalculation.Sum;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "CustomTableStyle";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from the FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    // Create data
    worksheet[1, 1].Text = "Products";
    worksheet[1, 2].Text = "Qtr1";
    worksheet[1, 3].Text = "Qtr2";
    worksheet[1, 4].Text = "Qtr3";
    worksheet[1, 5].Text = "Qtr4";
    worksheet[2, 1].Text = "Alfreds Futterkiste";
    worksheet[2, 2].Number = 744.6;
    worksheet[2, 3].Number = 162.56;
    worksheet[2, 4].Number = 5079.6;
    worksheet[2, 5].Number = 1249.2;
    worksheet[3, 1].Text = "Antonio Moreno";
    worksheet[3, 2].Number = 5079.6;
    worksheet[3, 3].Number = 1249.2;
    worksheet[3, 4].Number = 943.89;
    worksheet[3, 5].Number = 349.6;
    worksheet[4, 1].Text = "Around the Horn";
    worksheet[4, 2].Number = 1267.5;
    worksheet[4, 3].Number = 1062.5;
    worksheet[4, 4].Number = 744.6;
    worksheet[4, 5].Number = 162.56;
}

```

```

worksheet[5, 1].Text = "Bon app";
worksheet[5, 2].Number = 1418;
worksheet[5, 3].Number = 756;
worksheet[5, 4].Number = 1267.5;
worksheet[5, 5].Number = 1062.5;
worksheet[6, 1].Text = "Eastern Connection";
worksheet[6, 2].Number = 4728;
worksheet[6, 3].Number = 4547.92;
worksheet[6, 4].Number = 1418;
worksheet[6, 5].Number = 756;
worksheet[7, 1].Text = "Ernst Handel";
worksheet[7, 2].Number = 943.89;
worksheet[7, 3].Number = 349.6;
worksheet[7, 4].Number = 4728;
worksheet[7, 5].Number = 4547.92;
//Create style for table number format
IStyle style = workbook.Styles.Add("CurrencyFormat");
style.NumberFormat = "_($* #,##0.00_);_($* (#,##0.00);_($* \" -
\"??_);_(@_)\"";
worksheet["B2:E8"].CellStyleName = "CurrencyFormat";
//Create table
IListObject table = worksheet.ListObjects.Create("Table1",
worksheet["A1:E7"]);
//Apply custom table style
ITableStyles tableStyles = workbook.TableStyles;
ITableStyle tableStyle = tableStyles.Add("Table Style 1");
ITableStyleElements tableStyleElements = tableStyle.TableStyleElements;
ITableStyleElement tableStyleElement =
tableStyleElements.Add(ExcelTableStyleElementType.SecondColumnStripe);
tableStyleElement.BackColorRGB = Color.FromArgb(217, 225, 242);
ITableStyleElement tableStyleElement1 =
tableStyleElements.Add(ExcelTableStyleElementType.FirstColumn);
tableStyleElement1.FontColorRGB = Color.FromArgb(128, 128, 128);
ITableStyleElement tableStyleElement2 =
tableStyleElements.Add(ExcelTableStyleElementType.HeaderRow);
tableStyleElement2.FontColor = ExcelKnownColors.White;
tableStyleElement2.BackColorRGB = Color.FromArgb(0, 112, 192);
ITableStyleElement tableStyleElement3 =
tableStyleElements.Add(ExcelTableStyleElementType.TotalRow);
tableStyleElement3.BackColorRGB = Color.FromArgb(0, 112, 192);
tableStyleElement3.FontColor = ExcelKnownColors.White;
table.TableName = tableStyle.Name;
//Total row
table.ShowTotals = true;
table.ShowFirstColumn = true;
table.ShowTableStyleColumnStripes = true;
table.ShowTableStyleRowStripes = true;
table.Columns[0].TotalsRowLabel = "Total";
table.Columns[1].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[2].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[3].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[4].TotalsCalculation = ExcelTotalsCalculation.Sum;
//Saving the workbook as stream
FileStream stream = new FileStream("CustomTableStyle.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();

```



```
}

```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    // Create data
    worksheet[1, 1].Text = "Products";
    worksheet[1, 2].Text = "Qtr1";
    worksheet[1, 3].Text = "Qtr2";
    worksheet[1, 4].Text = "Qtr3";
    worksheet[1, 5].Text = "Qtr4";
    worksheet[2, 1].Text = "Alfreds Futterkiste";
    worksheet[2, 2].Number = 744.6;
    worksheet[2, 3].Number = 162.56;
    worksheet[2, 4].Number = 5079.6;
    worksheet[2, 5].Number = 1249.2;
    worksheet[3, 1].Text = "Antonio Moreno";
    worksheet[3, 2].Number = 5079.6;
    worksheet[3, 3].Number = 1249.2;
    worksheet[3, 4].Number = 943.89;
    worksheet[3, 5].Number = 349.6;
    worksheet[4, 1].Text = "Around the Horn";
    worksheet[4, 2].Number = 1267.5;
    worksheet[4, 3].Number = 1062.5;
    worksheet[4, 4].Number = 744.6;
    worksheet[4, 5].Number = 162.56;
    worksheet[5, 1].Text = "Bon app";
    worksheet[5, 2].Number = 1418;
    worksheet[5, 3].Number = 756;
    worksheet[5, 4].Number = 1267.5;
    worksheet[5, 5].Number = 1062.5;
    worksheet[6, 1].Text = "Eastern Connection";
    worksheet[6, 2].Number = 4728;
    worksheet[6, 3].Number = 4547.92;
    worksheet[6, 4].Number = 1418;
    worksheet[6, 5].Number = 756;
    worksheet[7, 1].Text = "Ernst Handel";
    worksheet[7, 2].Number = 943.89;
    worksheet[7, 3].Number = 349.6;
    worksheet[7, 4].Number = 4728;
    worksheet[7, 5].Number = 4547.92;
    //Create style for table number format
    IStyle style = workbook.Styles.Add("CurrencyFormat");
    style.NumberFormat = "_($* #,##0.00_);_($* (#,##0.00);_($* \" - \"?_?_);_(@_)"
    worksheet["B2:E8"].CellStyleName = "CurrencyFormat";
    //Create table
    IListObject table = worksheet.ListObjects.Create("Table1",
    worksheet["A1:E7"]);
    //Apply custom table style
    ITableStyles tableStyles = workbook.TableStyles;

```

```

ITableStyle tableStyle = tableStyles.Add("Table Style 1");
ITableStyleElements tableStyleElements = tableStyle.TableStyleElements;
ITableStyleElement tableStyleElement =
tableStyleElements.Add(ExcelTableStyleElementType.SecondColumnStripe);
tableStyleElement.BackColorRGB = Color.FromArgb(217, 225, 242);
ITableStyleElement tableStyleElement1 =
tableStyleElements.Add(ExcelTableStyleElementType.FirstColumn);
tableStyleElement1.FontColorRGB = Color.FromArgb(128, 128, 128);
ITableStyleElement tableStyleElement2 =
tableStyleElements.Add(ExcelTableStyleElementType.HeaderRow);
tableStyleElement2.FontColor = ExcelKnownColors.White;
tableStyleElement2.BackColorRGB = Color.FromArgb(0, 112, 192);
ITableStyleElement tableStyleElement3 =
tableStyleElements.Add(ExcelTableStyleElementType.TotalRow);
tableStyleElement3.BackColorRGB = Color.FromArgb(0, 112, 192);
tableStyleElement3.FontColor = ExcelKnownColors.White;
table.TableName = tableStyle.Name;
//Total row
table.ShowTotals = true;
table.ShowFirstColumn = true;
table.ShowTableStyleColumnStripes = true;
table.ShowTableStyleRowStripes = true;
table.Columns[0].TotalsRowLabel = "Total";
table.Columns[1].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[2].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[3].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[4].TotalsCalculation = ExcelTotalsCalculation.Sum;
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS
platforms. Please refer xlsio/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Custom
TableStyle.xlsx",
"application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("CustomTableStyle.x
lsx",
"application/msexcel", stream);
}
}

```

The following screenshot represents generated Excel file with custom table styles in XlsIO.

Products	Qtr1	Qtr2	Qtr3	Qtr4
Alfreds Futterkiste	\$ 744.60	\$ 162.56	\$ 5,079.60	\$ 1,249.20
Antonio Moreno Taqueria	\$ 5,079.60	\$ 1,249.20	\$ 943.89	\$ 349.60
Around the Horn	\$ 1,267.50	\$ 1,062.50	\$ 744.60	\$ 162.56
Bon app	\$ 1,418.00	\$ 756.00	\$ 1,267.50	\$ 1,062.50
Eastern Connection	\$ 4,728.00	\$ 4,547.92	\$ 1,418.00	\$ 756.00
Ernst Handel	\$ 943.89	\$ 349.60	\$ 4,728.00	\$ 4,547.92
Total	\$ 14,181.59	\$ 8,127.78	\$ 14,181.59	\$ 8,127.78

Insert or remove columns in a table

IListObject is a collection of columns, whereas a single column is represented by an instance of **IListObjectColumn**. XlsIO supports to insert or remove columns from the table using worksheet, as follows.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating a table
    IListObject table = worksheet.ListObjects.Create("Table1",
        worksheet["A1:C8"]);
    //Inserting a column in the table
    worksheet.InsertColumn(2, 2);
    // Removing a column from the table
    worksheet.DeleteColumn(2, 1);
    string fileName = "Output.xlsx";
    workbook.SaveAs(fileName);
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
    ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Creating table
Dim table As IListObject = worksheet.ListObjects.Create("Table1",
    worksheet("A1:C8"))
'Inserting a column in the table
worksheet.InsertColumn(2, 2)
'Removing a column from the table
worksheet.DeleteColumn(2, 1)
Dim fileName As String = "Output.xlsx"
workbook.SaveAs(fileName)
End Using
```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("Table.Sample.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating a table
    IListObject table = worksheet.ListObjects.Create("Table1",
        worksheet["A1:C8"]);
    //Inserting a column in the table
    worksheet.InsertColumn(2, 2);
    //Removing a column from the table
    worksheet.DeleteColumn(2, 1);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from the FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream,
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating a table
    IListObject table = worksheet.ListObjects.Create("Table1",
        worksheet["A1:C8"]);
    //Inserting a column in the table
    worksheet.InsertColumn(2, 2);
    //Removing a column from the table
    worksheet.DeleteColumn(2, 1);
    string fileName = "Output.xlsx";
    //Saving the workbook as stream
    FileStream stream = new FileStream(fileName, FileMode.Create,
        FileAccess.ReadWrite);
    workbook.SaveAs(stream);
}

```

```
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("Table.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating a table
    IListObject table = worksheet.ListObjects.Create("Table1",
        worksheet["A1:C8"]);
    //Inserting a column in the table
    worksheet.InsertColumn(2, 2);
    //Removing a column from the table
    worksheet.DeleteColumn(2, 1);
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
    string fileName = "Output.xlsx";
    outputStream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
            "application/msexcel", outputStream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
            "application/msexcel", outputStream);
    }
}
```

Note: Inserting rows or columns in a worksheet within the table range modifies table structure.

Adding a total row

The "Total Row" is added to a table by accessing the **Table Columns**. It is possible to set calculation function to be used to the total row cells by using the **ExcelTotalsCalculation** enumerator. To learn more about this enumerator, refer to the **ExcelTotalsCalculation** in API section. These cells are updated after they are calculated.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating a table
    IListObject table = worksheet.ListObjects.Create("Table1",
        worksheet["A1:C8"]);
    //Adding total row
    table.ShowTotals = true;
    table.Columns[0].TotalsRowLabel = "Total";
    table.Columns[1].TotalsCalculation = ExcelTotalsCalculation.Sum;
    table.Columns[2].TotalsCalculation = ExcelTotalsCalculation.Sum;
    string fileName = "Output.xlsx";
    workbook.SaveAs(fileName);
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
    ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Creating a table
Dim table As IListObject = worksheet.ListObjects.Create("Table1",
    worksheet("A1:C8"))
'Adding total row
table.ShowTotals = True
table.Columns(0).TotalsRowLabel = "Total"
table.Columns(1).TotalsCalculation = ExcelTotalsCalculation.Sum
table.Columns(2).TotalsCalculation = ExcelTotalsCalculation.Sum
Dim fileName As String = "Output.xlsx"
workbook.SaveAs(fileName)
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("Table.Sample.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating a table
    IListObject table = worksheet.ListObjects.Create("Table1",
        worksheet["A1:C8"]);
}

```

```

//Adding total row
table.ShowTotals = true;
table.Columns[0].TotalsRowLabel = "Total";
table.Columns[1].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[2].TotalsCalculation = ExcelTotalsCalculation.Sum;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream,
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating a table
    IListObject table = worksheet.ListObjects.Create("Table1",
    worksheet["A1:C8"]);
    //Adding Total Row
    table.ShowTotals = true;
    table.Columns[0].TotalsRowLabel = "Total";
    table.Columns[1].TotalsCalculation = ExcelTotalsCalculation.Sum;
    table.Columns[2].TotalsCalculation = ExcelTotalsCalculation.Sum;
    string fileName = "Output.xlsx";
    //Saving the workbook as stream
    FileStream stream = new FileStream(fileName, FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("Table.Sample.xlsx");
}

```

```

IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Creating a table
IListObject table = worksheet.ListObjects.Create("Table1",
worksheet["A1:C8"]);
//Adding total row
table.ShowTotals = true;
table.Columns[0].TotalsRowLabel = "Total";
table.Columns[1].TotalsCalculation = ExcelTotalsCalculation.Sum;
table.Columns[2].TotalsCalculation = ExcelTotalsCalculation.Sum;
//Saving the workbook as stream
MemoryStream outputStream = new MemoryStream();
workbook.SaveAs(outputStream);
string fileName = "Output.xlsx";
outputStream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies among Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
"application/msexcel", outputStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
"application/msexcel", outputStream);
}
}

```

Create a table from external connection

External connection support allows to work with most recent data right in the workbook. After the data is imported, only refresh operations are performed to retrieve the updated data.

The following code snippet explains the method of importing data through an external connection in the workbook.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Database path
string dataPath = Path.GetFullPath(@"c:\company\DB\TestDB.mdb");
//Connection string for DataSource
string ConnectionString =
"OLEDB;Provider=Microsoft.JET.OLEDB.4.0;Password=\"\";User ID=Admin;Data
Source=" + dataPath;
//Adding a connection to the workbook

```



```

IConnection Connection = workbook.Connections.Add("Connection1", "Sample
connection with MsAccess", ConnectionString, "", ExcelCommandType.Sql);
//Adding a QueryTable to sheet object
worksheet.ListObjects.AddEx(ExcelListObjectSourceType.SrcQuery, Connection,
worksheet.Range["A1"]);
//Command text for the connection
worksheet.ListObjects[0].QueryTable.CommandText = "Select * from tableTest";
//The query performs asynchronous action
worksheet.ListObjects[0].QueryTable.BackgroundQuery = true;
//The query table is refreshed when the workbook is opened
worksheet.ListObjects[0].QueryTable.RefreshOnFileOpen = true;
//Represents the connection description
Connection.Description = "Sample Connection";
//Import data to the sheet from the database
worksheet.ListObjects[0].Refresh();
//Auto-fits the columns
worksheet.UsedRange.AutoFitColumns();
string fileName = "Output.xlsx";
workbook.SaveAs(fileName);
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Database path
Dim dataPath As String = Path.GetFullPath("c:\company\DB\TestDB.mdb")
'Connection string for DataSource
Dim ConnectionString As String =
"OLEDB;Provider=Microsoft.JET.OLEDB.4.0;Password=""";User ID=Admin;Data
Source=" + dataPath
'Adding a connection to the workbook
Dim Connection As IConnection = workbook.Connections.Add("Connection1",
"Sample connection with MsAccess", ConnectionString, "",
ExcelCommandType.Sql)
'Adding a QueryTable to sheet object
worksheet.ListObjects.AddEx(ExcelListObjectSourceType.SrcQuery, Connection,
worksheet.Range("A1"))
'Command text for the connection
worksheet.ListObjects(0).QueryTable.CommandText = "Select * from tableTest"
'The query performs asynchronous action
worksheet.ListObjects(0).QueryTable.BackgroundQuery = True
'The QueryTable is refreshed when the workbook is opened
worksheet.ListObjects(0).QueryTable.RefreshOnFileOpen = True
'Represents the connection description
Connection.Description = "Sample Connection"
'Import data to the sheet from the database
worksheet.ListObjects(0).Refresh()
'Auto-fits the columns
worksheet.UsedRange.AutoFitColumns()
Dim fileName As String = "Output.xlsx"
workbook.SaveAs(fileName)
End Using

```

UWP

```
//XlsIO supports creation of table from external connection in Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms.
```

ASP.NET CORE

```
//XlsIO supports creation of table from external connection in Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms.
```

XAMARIN

```
//XlsIO supports creation of table from external connection in Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms.
```

The following table shows different data sources and its connection string formats supported in XlsIO.

Database	Connection Type	Sample connection string
Microsoft Access	OleDb	OleDb;Provider=Microsoft.JET.OleDb.4.0;Password=""; User ID=Admin;Data Source=C:\\Company\\DB\\TestDB.mdb
	ODBC	ODBC;DSN=MS Access;DBQ=C:\\Company\\DB\\Testing.mdb; DefaultDir=C:\\Company\\DB;FIL=MS Access;MaxBufferSize=2048;PageTimeout=5;
SQL	OleDb	OleDb;Provider=SQLOLEDB.1;Integrated Security=SSPI; Persist Security Info=True;Initial Catalog=Temp; Data Source=SYNCFUSION\\SQLEXPRESS;Workstation ID=SYNCCINC;
	ODBC	ODBC;DSN=Test1;UID=syncfusion;Trusted_Connection=Yes; APP=Microsoft Office 2010;WSID=SYNCCINC;DATABASE=Temp
Excel	OleDb	OleDb;Provider=Microsoft.ACE.OleDb.12.0;Password=""; User ID=Admin;Data Source="c:\\SourceTemplate.xlsx; Jet OleDb:Engine Type=37;

SharePoint	OleDb	Stars with OleDb
	Odbc	Stars with Odbc

Adding parameters to query in Excel table

Excel tables can be created by importing data from SQL Server through Excel data connections. The queries used to fetch data from SQL Server can be modified at run-time with the help of its parameters. There are three types of parameters which are applied to the WHERE clause of the SQL query. Let's see the types in detail and how to implement them.

Note: Excel table must be refreshed to obtain the filtered result with parameters. Table refresh operation is not supported in UWP, ASP.NET Core and Xamarin platforms.

Set parameter as Prompt

The following code example illustrates how to set parameter through prompt event.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("QueryTable.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Get query table from list objects.
    QueryTableImpl queryTable = worksheet.ListObjects[0].QueryTable;
    //Set SQL query to the query table Add parameters to the query table.
    queryTable.CommandText = "select * from Employee_Details where Emp_Age >
?;";
    //Add parameters to the query table.
    IParameter parameter = queryTable.Parameters.Add("parameter1",
ExcelParameterDataType.SQLSmallInt);
    //Set parameter value through prompt.
    parameter.SetParam(ExcelParameterType.Prompt, "Prompt");
    //Set prompt event handler to update parameter value.
    parameter.Prompt += new PromptEventHandler(SetParameter);
    //Refresh the listobject
    worksheet.ListObjects[0].Refresh();
    workbook.SaveAs("PromptParameter.xlsx");
    workbook.Close();
}

private void SetParameter(object sender, PromptEventArgs args)
{
    args.Value = 20;
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("QueryTable.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Get query table from list objects.
Dim queryTable As QueryTableImpl = worksheet.ListObjects(0).QueryTable
```

```

'Set SQL query to the query table Add parameters to the query table.
queryTable.CommandText = "select * from Employee_Details where Emp_Age > ?;"
'Add parameters to the query table.
Dim parameter As IParameter = queryTable.Parameters.Add("parameter1",
ExcelParameterDataType.SQLSmallInt)
'Set parameter value through prompt.
parameter.SetParam(ExcelParameterType.Prompt, "Prompt")
'Set prompt event handler to update parameter value
parameter.Prompt += New PromptEventHandler(SetParameter)
'Refresh the listobject
worksheet.ListObjects(0).Refresh()
workbook.SaveAs("PromptParameter.xlsx")
workbook.Close()
End Using
Private Sub SetParameter(ByVal sender As Object, ByVal args As
PromptEventArgs)
args.Value = 20
End Sub

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from embedded resource collection
Stream inputStream =
assembly.GetManifestResourceStream("Sample.QueryTable.xlsx");
IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Get query table from list objects.
QueryTableImpl queryTable = worksheet.ListObjects[0].QueryTable;
//Set SQL query to the query table Add parameters to the query table.
queryTable.CommandText = "select * from Employee_Details where Emp_Age >
?;";
//Add parameters to the query table.
IParameter parameter = queryTable.Parameters.Add("parameter1",
ExcelParameterDataType.SQLSmallInt);
//Set parameter value through prompt.
parameter.SetParam(ExcelParameterType.Prompt, "Prompt");
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "PromptParameter";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("QueryTable.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream,
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    ///Get query table from list objects.
    QueryTableImpl queryTable = worksheet.ListObjects[0].QueryTable;
    ///Set SQL query to the query table Add parameters to the query table.
    queryTable.CommandText = "select * from Employee_Details where Emp_Age >
    ?;";
    ///Add parameters to the query table.
    IParameter parameter = queryTable.Parameters.Add("parameter1",
    ExcelParameterDataType.SQLSmallInt);
    ///Set parameter value through prompt.
    parameter.SetParam(ExcelParameterType.Prompt, "Prompt");
    ///Saving the workbook as stream
    FileStream stream = new FileStream("PromptParameter.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    ///Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    ///Gets input Excel document from embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("Sample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    ///Get query table from list objects.
    QueryTableImpl queryTable = worksheet.ListObjects[0].QueryTable;
    ///Set SQL query to the query table Add parameters to the query table.
    queryTable.CommandText = "select * from Employee_Details where Emp_Age >
    ?;";
    ///Add parameters to the query table.
    IParameter parameter = queryTable.Parameters.Add("parameter1",
    ExcelParameterDataType.SQLSmallInt);
    ///Set parameter value through prompt.
    parameter.SetParam(ExcelParameterType.Prompt, "Prompt");
    ///Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
    outputStream.Position = 0;
    ///Save the document as file and view the saved document
    ///The operation in SaveAndView under Xamarin varies among Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
}

```

```

if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Prompt
Parameter.xlsx", "application/msexcel", outputStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("PromptParameter.xl
sx", "application/msexcel", outputStream);
}
}

```

Set parameter as Constant

The following code example illustrates how to set parameter through *constant* type.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("QueryTable.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
//Get query table from list objects.
QueryTableImpl queryTable = worksheet.ListObjects[0].QueryTable;
//Set SQL query to the query table Add parameters to the query table.
queryTable.CommandText = "select * from Employee_Details where Emp_Age <
?;";
//Add parameters to the query table.
IParameter parameter = queryTable.Parameters.Add("parameter1",
ExcelParameterDataType.SQLSmallInt);
//Set constant to the parameter value.
parameter.SetParam(ExcelParameterType.Constant, 30);
//Refresh the listobject
worksheet.ListObjects[0].Refresh();
workbook.SaveAs("ConstantParameter.xlsx");
workbook.Close();
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("QueryTable.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Get query table from list objects.
Dim queryTable As QueryTableImpl = worksheet.ListObjects(0).QueryTable
'Set SQL query to the query table Add parameters to the query table.
queryTable.CommandText = "select * from Employee_Details where Emp_Age < ?;";
'Add parameters to the query table.
Dim parameter As IParameter = queryTable.Parameters.Add("parameter1",
ExcelParameterDataType.SQLSmallInt)
'Set constant to the parameter value.
parameter.SetParam(ExcelParameterType.Constant, 30)
'Refresh the listobject
worksheet.ListObjects(0).Refresh()

```

```
workbook.SaveAs("ConstantParameter.xlsx")
workbook.Close()
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("Sample.QueryTable.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Get query table from list objects.
    QueryTableImpl queryTable = worksheet.ListObjects[0].QueryTable;
    //Set SQL query to the query table Add parameters to the query table.
    queryTable.CommandText = "select * from Employee_Details where Emp_Age <
    ?;";
    //Add parameters to the query table.
    IParameter parameter = queryTable.Parameters.Add("parameter1",
        ExcelParameterDataType.SQLSmallInt);
    //Set constant to the parameter value.
    parameter.SetParam(ExcelParameterType.Constant, 30);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "ConstantParameter";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("QueryTable.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream,
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    ///Get query table from list objects.
    QueryTableImpl queryTable = worksheet.ListObjects[0].QueryTable;
    //Set SQL query to the query table Add parameters to the query table.
    queryTable.CommandText = "select * from Employee_Details where Emp_Age <
    ?;";
    //Add parameters to the query table.
```

```

IParameter parameter = queryTable.Parameters.Add("parameter1",
ExcelParameterDataType.SQLSmallInt);
//Set constant to the parameter value.
parameter.SetParam(ExcelParameterType.Constant, 30);
//Saving the workbook as stream
FileStream stream = new FileStream("ConstantParameter.xlsx",
FileMode.Create, FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
assembly.GetManifestResourceStream("Sample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    ///Get query table from list objects.
    QueryTableImpl queryTable = worksheet.ListObjects[0].QueryTable;
    //Set SQL query to the query table Add parameters to the query table.
    queryTable.CommandText = "select * from Employee_Details where Emp_Age <
?;";
    //Add parameters to the query table.
    IParameter parameter = queryTable.Parameters.Add("parameter1",
ExcelParameterDataType.SQLSmallInt);
    //Set constant to the parameter value.
    parameter.SetParam(ExcelParameterType.Constant, 30);
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
    outputStream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Consta
ntParameter.xlsx", "application/msexcel", outputStream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ConstantParameter.
xlsx", "application/msexcel", outputStream);
    }
}

```


Set parameter as Range

The following code example illustrates how to set parameter type to a specific range.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("QueryTable.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Get query table from list objects.
    QueryTableImpl queryTable = worksheet.ListObjects[0].QueryTable;
    //Set SQL query to the query table Add parameters to the query table.
    queryTable.CommandText = "select * from Employee_Details where Emp_Age > ?;";
    //Add parameters to the query table.
    IParameter parameter = queryTable.Parameters.Add("parameter1",
        ExcelParameterDataType.SQLSmallInt);
    //Set range to the parameter value.
    parameter.SetParam(ExcelParameterType.Range, worksheet.Range["H1"]);
    //Refresh the listobject
    worksheet.ListObjects[0].Refresh();
    workbook.SaveAs("RangeParameter.xlsx");
    workbook.Close();
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("QueryTable.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Get query table from list objects.
Dim queryTable As QueryTableImpl = worksheet.ListObjects(0).QueryTable
'Set SQL query to the query table Add parameters to the query table.
queryTable.CommandText = "select * from Employee_Details where Emp_Age > ?;";
'Add parameters to the query table.
Dim parameter As IParameter = queryTable.Parameters.Add("parameter1",
    ExcelParameterDataType.SQLSmallInt)
'Set range to the parameter value.
parameter.SetParam(ExcelParameterType.Range, worksheet.Range["H1"])
'Refresh the listobject
worksheet.ListObjects(0).Refresh()
workbook.SaveAs("RangeParameter.xlsx")
workbook.Close()
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
```

```

Stream inputStream =
assembly.GetManifestResourceStream("Sample.QueryTable.xlsx");
IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Get query table from list objects.
QueryTableImpl queryTable = worksheet.ListObjects[0].QueryTable;
//Set SQL query to the query table Add parameters to the query table.
queryTable.CommandText = "select * from Employee_Details where Emp_Age >
?;";
//Add parameters to the query table.
IParameter parameter = queryTable.Parameters.Add("parameter1",
ExcelParameterDataType.SQLSmallInt);
//Set range to the parameter value.
parameter.SetParam(ExcelParameterType.Range, worksheet.Range["H1"]);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "RangeParameter";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
FileStream fileStream = new FileStream("QueryTable.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
///Get query table from list objects.
QueryTableImpl queryTable = worksheet.ListObjects[0].QueryTable;
//Set SQL query to the query table Add parameters to the query table.
queryTable.CommandText = "select * from Employee_Details where Emp_Age >
?;";
//Add parameters to the query table.
IParameter parameter = queryTable.Parameters.Add("parameter1",
ExcelParameterDataType.SQLSmallInt);
//Set range to the parameter value.
parameter.SetParam(ExcelParameterType.Range, worksheet.Range["H1"]);
//Saving the workbook as stream
FileStream stream = new FileStream("RangeParameter.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("Sample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    ///Get query table from list objects.
    QueryTableImpl queryTable = worksheet.ListObjects[0].QueryTable;
    //Set SQL query to the query table Add parameters to the query table.
    queryTable.CommandText = "select * from Employee_Details where Emp_Age >
        ?;";
    //Add parameters to the query table.
    IParameter parameter = queryTable.Parameters.Add("parameter1",
        ExcelParameterDataType.SQLSmallInt);
    //Set range to the parameter value.
    parameter.SetParam(ExcelParameterType.Range, worksheet.Range["H1"]);
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
    outputStream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("RangeP
            arameter.xlsx", "application/msexcel", outputStream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("RangeParameter.xls
            x", "application/msexcel", outputStream);
    }
}

```

Working with Pictures

XlsIO allows to insert Pictures into a worksheet. Refer to the following code snippet.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding a picture
    IPictureShape shape = worksheet.Pictures.AddPicture(1, 1, "Image.png");
    workbook.SaveAs("AddingImage.xlsx");
}

```

```
}

```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding a picture
Dim shape As IPictureShape = worksheet.Pictures.AddPicture(1, 1,
"Image.png")
workbook.SaveAs("AddingImage.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Adding a picture
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream imageStream =
assembly.GetManifestResourceStream("UWP.Data.Image.png");
IPictureShape shape = worksheet.Pictures.AddPicture(1, 1, imageStream);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "AddingImage";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Adding a picture
FileStream imageStream = new FileStream("Image.png", FileMode.Open,
FileAccess.Read);
IPictureShape shape = worksheet.Pictures.AddPicture(1, 1, imageStream);
//Saving the workbook as stream
FileStream stream = new FileStream("AddingImage.xlsx", FileMode.Create,
FileAccess.ReadWrite);
```

```
workbook.SaveAs(stream);
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding a picture
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream imageStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Image.png");
    IPictureShape shape = worksheet.Pictures.AddPicture(1, 1, imageStream);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Adding
        Image.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("AddingImage.xlsx",
        "application/msexcel", stream);
    }
}
```

Positioning and Re-Sizing Pictures

Pictures can be re-sized, positioned, and formatted using various properties of **IPictureShape** interface. Refer to the following code snippet.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding a Picture
    IPictureShape shape = worksheet.Pictures.AddPicture(1, 1, "Image.png");
    //Positioning a Picture
    shape.Top = 100;
```

```

shape.Left = 100;
//Re-sizing a Picture
shape.Height = 100;
shape.Width = 100;
workbook.SaveAs("AddingImage.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding a Picture
Dim shape As IPictureShape = worksheet.Pictures.AddPicture(1, 1,
"Image.png")
'Positioning a Picture
shape.Top = 100
shape.Left = 100
'Re-sizing a Picture
shape.Height = 100
shape.Width = 100
workbook.SaveAs("AddingImage.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Adding a picture
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream imageStream =
assembly.GetManifestResourceStream("UWP.Data.Image.png");
IPictureShape shape = worksheet.Pictures.AddPicture(1, 1, imageStream);
//Positioning a Picture
shape.Top = 100;
shape.Left = 100;
//Re-sizing a Picture
shape.Height = 100;
shape.Width = 100;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "AddingImage";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding a picture
    FileStream imageStream = new FileStream("Image.png", FileMode.Open,
    FileAccess.Read);
    IPictureShape shape = worksheet.Pictures.AddPicture(1, 1, imageStream);
    //Positioning a Picture
    shape.Top = 100;
    shape.Left = 100;
    //Re-sizing a Picture
    shape.Height = 100;
    shape.Width = 100;
    //Saving the workbook as stream
    FileStream stream = new FileStream("AddingImage.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding a picture
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream imageStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Ima
    ge.png");
    IPictureShape shape = worksheet.Pictures.AddPicture(1, 1, imageStream);
    //Positioning a Picture
    shape.Top = 100;
    shape.Left = 100;
    //Re-sizing a Picture
    shape.Height = 100;
    shape.Width = 100;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)

```

```
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Adding
Image.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("AddingImage.xlsx",
"application/msexcel", stream);
}
}
```

Adding Images from External Link

An image can be added to a worksheet as an external link without downloading the original image. The picture will be downloaded every time the spreadsheet is opened in Microsoft Excel. The image is not physically embedded into the Excel document, but points to a web resource. Refer to the following code snippet.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Add image from the specified url at the specified location in the
worksheet
worksheet.Pictures.AddPictureAsLink(1, 1, 5, 7,
"https://cdn.syncfusion.com/content/images/company-
logos/Syncfusion_Logo_Image.png");
//Save workbook
workbook.SaveAs("ExternalImage.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Add image from the specified url at the specified location in the worksheet
worksheet.Pictures.AddPictureAsLink(1, 1, 5, 7,
"https://cdn.syncfusion.com/content/images/company-
logos/Syncfusion_Logo_Image.png")
'Save workbook
workbook.SaveAs("ExternalImage.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
```



```

application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Add image from the specified url at the specified location in the
worksheet
worksheet.Pictures.AddPictureAsLink(1, 1, 5, 7,
"https://cdn.syncfusion.com/content/images/company-
logos/Syncfusion_Logo_Image.png");
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ExternalImage";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Add image from the specified url at the specified location in the
worksheet
worksheet.Pictures.AddPictureAsLink(1, 1, 5, 7,
"https://cdn.syncfusion.com/content/images/company-
logos/Syncfusion_Logo_Image.png");
//Saving the workbook as stream
FileStream stream = new FileStream("ExternalImage.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Add image from the specified url at the specified location in the
worksheet
worksheet.Pictures.AddPictureAsLink(1, 1, 5, 7,
"https://cdn.syncfusion.com/content/images/company-
logos/Syncfusion_Logo_Image.png");
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
}

```

```

stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android, and iOS platforms. Refer to the xlsio/xamarin section for
//respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("ExternalImage.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ExternalImage.xlsx", "application/msexcel", stream);
}
}

```

Adding SVG Images

SVG images can be inserted in Excel documents for displaying images with accuracy when scaling or zooming page. Adding SVG images is now supported in XlsIO with SVG and its fallback raster image data as parameters as in the following code snippet.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
FileStream svgStream = new FileStream("Sample.svg", FileMode.Open);
FileStream pngStream = new FileStream("Sample.png", FileMode.Open);
//Add svg image with given svg and png streams
worksheet.Pictures.AddPicture(1, 1, svgStream, pngStream);
//Save workbook
workbook.SaveAs("Svg.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim svgStream As New FileStream("Sample.svg", FileMode.Open)
Dim pngStream As New FileStream("Sample.png", FileMode.Open)
'Add svg image with given svg and png streams
worksheet.Pictures.AddPicture(1, 1, svgStream, pngStream)
'Save workbook
workbook.SaveAs("Svg.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream svgStream =
    assembly.GetManifestResourceStream("UWP.Data.Sample.svg");
    Stream pngStream =
    assembly.GetManifestResourceStream("UWP.Data.Sample.png");
    ///Add svg image with given svg and png streams
    worksheet.Pictures.AddPicture(1, 1, svgStream, pngStream);
    ///Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Svg";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    ///Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    ///Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    FileStream svgStream = new FileStream("Sample.svg", FileMode.Open);
    FileStream pngStream = new FileStream("Sample.png", FileMode.Open);
    ///Add svg image with given svg and png streams
    worksheet.Pictures.AddPicture(1, 1, svgStream, pngStream);
    ///Saving the workbook as stream
    FileStream stream = new FileStream("Svg.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    ///"App" is the class of Portable project

```

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream svgStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.svg");
Stream pngStream =
assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.png");
//Add svg image with given svg and png streams
worksheet.Pictures.AddPicture(1, 1, svgStream, pngStream);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android, and iOS platforms. Refer to the xlsio/xamarin section for
//respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Svg.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Svg.xlsx", "application/msexcel", stream);
}
}

```

Working with Pivot Tables

You can easily arrange and summarize complex data in a [Pivot Table](#). Creation and Manipulation of pivot table is supported in Excel 2007 and later formats, and pivot table in an existing document can be preserved for Excel 2003 format.

Create a pivot table

Steps to create a simple pivot table:

- Add pivot cache
- Add pivot table
- Add row and column fields
- Add data fields

Pivot tables do not take data directly from the source data, but take from the pivot cache that memorizes a snapshot of the data. The **IPivotCache** interface caches the data that needs to be summarized.

The data in worksheet is added to the pivot cache as follows.

C#

```

//Create Pivot cache with the given data range
IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:H50"]);

```

VB.NET

```
'Create Pivot cache with the given data range  
Dim cache As IPivotCache = workbook.PivotCaches.Add(worksheet("A1:H50"))
```

UWP

```
//Create Pivot cache with the given data range  
IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:H50"]);
```

ASP.NET CORE

```
//Create Pivot cache with the given data range  
IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:H50"]);
```

XAMARIN

```
//Create Pivot cache with the given data range  
IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:H50"]);
```

IPivotTable represents a single pivot table object created from the cache. It has properties that customizes the pivot table. The following code creates a blank pivot table.

C#

```
//Create "PivotTable1" with the cache at the specified range  
IPivotTable pivotTable = worksheet.PivotTables.Add("PivotTable1",  
worksheet["A1"], cache);
```

VB.NET

```
'Create "PivotTable1" with the cache at the specified range  
Dim pivotTable As IPivotTable = worksheet.PivotTables.Add("PivotTable1",  
worksheet("A1"), cache)
```

UWP

```
//Create "PivotTable1" with the cache at the specified range  
IPivotTable pivotTable = worksheet.PivotTables.Add("PivotTable1",  
worksheet["A1"], cache);
```

ASP.NET CORE

```
//Create "PivotTable1" with the cache at the specified range  
IPivotTable pivotTable = worksheet.PivotTables.Add("PivotTable1",  
worksheet["A1"], cache);
```

XAMARIN

```
//Create "PivotTable1" with the cache at the specified range  
IPivotTable pivotTable = worksheet.PivotTables.Add("PivotTable1",  
worksheet["A1"], cache);
```

The pivot table should be populated with required fields. The **IPivotField** represents a single field in the pivot table, which includes row, column, and data field axes.

C#

```
//Add Pivot table fields (row and column fields)
pivotTable.Fields[2].Axis = PivotAxisTypes.Row;
pivotTable.Fields[6].Axis = PivotAxisTypes.Row;
pivotTable.Fields[3].Axis = PivotAxisTypes.Column;
```

VB.NET

```
'Add Pivot table fields (row and column fields)
pivotTable.Fields(2).Axis = PivotAxisTypes.Row
pivotTable.Fields(6).Axis = PivotAxisTypes.Row
pivotTable.Fields(3).Axis = PivotAxisTypes.Column
```

UWP

```
//Add Pivot table fields (row and column fields)
pivotTable.Fields[2].Axis = PivotAxisTypes.Row;
pivotTable.Fields[6].Axis = PivotAxisTypes.Row;
pivotTable.Fields[3].Axis = PivotAxisTypes.Column;
```

ASP.NET CORE

```
//Add Pivot table fields (row and column fields)
pivotTable.Fields[2].Axis = PivotAxisTypes.Row;
pivotTable.Fields[6].Axis = PivotAxisTypes.Row;
pivotTable.Fields[3].Axis = PivotAxisTypes.Column;
```

XAMARIN

```
//Add Pivot table fields (row and column fields)
pivotTable.Fields[2].Axis = PivotAxisTypes.Row;
pivotTable.Fields[6].Axis = PivotAxisTypes.Row;
pivotTable.Fields[3].Axis = PivotAxisTypes.Column;
```

The **IPivotDataFields** represents a collection of data fields in the pivot table. The data field is added with the required subtotal function using the **PivotSubtotalTypes** enumeration. To learn more about different subtotal types supported in pivot tables, refer to the **PivotSubtotalTypes** in API section. The following code explains how to configure a pivot field as a data field.

C#

```
//Add data field
IPivotField field = pivotTable.Fields[5];
pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum);
```

VB.NET

```
'Add data field
Dim field As IPivotField = pivotTable.Fields(5)
pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum)
```

UWP

```
//Add data field
IPivotField field = pivotTable.Fields[5];
pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum);
```

ASP.NET CORE

```
//Add data field
IPivotField field = pivotTable.Fields[5];
pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum);
```

XAMARIN

```
//Add data field
IPivotField field = pivotTable.Fields[5];
pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum);
```

The following code snippet illustrates how to create a pivot table with existing data in the worksheet using XlsIO.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("PivotData.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    IWorksheet pivotSheet = workbook.Worksheets[1];
    //Create Pivot cache with the given data range
    IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:H50"]);
    //Create "PivotTable1" with the cache at the specified range
    IPivotTable pivotTable = pivotSheet.PivotTables.Add("PivotTable1",
    pivotSheet["A1"], cache);
    //Add Pivot table fields (Row and Column fields)
    pivotTable.Fields[2].Axis = PivotAxisTypes.Row;
    pivotTable.Fields[6].Axis = PivotAxisTypes.Row;
    pivotTable.Fields[3].Axis = PivotAxisTypes.Column;
    //Add data field
    IPivotField field = pivotTable.Fields[2];
    pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum);
    workbook.SaveAs("PivotTable.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("PivotData.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim pivotSheet As IWorksheet = workbook.Worksheets(1)
```

```

'Create Pivot cache with the given data range
Dim cache As IPivotCache = workbook.PivotCaches.Add(worksheet("A1:H50"))
'Create "PivotTable1" with the cache at the specified range
Dim pivotTable As IPivotTable = pivotSheet.PivotTables.Add("PivotTable1",
pivotSheet("A1"), cache)
'Add Pivot table fields (Row and Column fields)
pivotTable.Fields(2).Axis = PivotAxisTypes.Row
pivotTable.Fields(6).Axis = PivotAxisTypes.Row
pivotTable.Fields(3).Axis = PivotAxisTypes.Column
'Add data field
Dim field As IPivotField = pivotTable.Fields(2)
pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum)
workbook.SaveAs("PivotTable.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
assembly.GetManifestResourceStream("PivotTable.PivotData.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    IWorksheet pivotSheet = workbook.Worksheets[1];
    //Create Pivot cache with the given data range
    IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:H50"]);
    //Create "PivotTable1" with the cache at the specified range
    IPivotTable pivotTable = pivotSheet.PivotTables.Add("PivotTable1",
pivotSheet["A1"], cache);
    //Add Pivot table fields (Row and Column fields)
    pivotTable.Fields[2].Axis = PivotAxisTypes.Row;
    pivotTable.Fields[6].Axis = PivotAxisTypes.Row;
    pivotTable.Fields[3].Axis = PivotAxisTypes.Column;
    //Add data field
    IPivotField field = pivotTable.Fields[2];
    pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "PivotTable";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())

```



```

{
    IApplication application = excelEngine.Excel;
    FileStream fileStream = new FileStream("PivotData.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    IWorksheet pivotSheet = workbook.Worksheets[1];
    //Create Pivot cache with the given data range
    IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:H50"]);
    //Create "PivotTable1" with the cache at the specified range
    IPivotTable pivotTable = pivotSheet.PivotTables.Add("PivotTable1",
    pivotSheet["A1"], cache);
    //Add Pivot table fields (Row and Column fields)
    pivotTable.Fields[2].Axis = PivotAxisTypes.Row;
    pivotTable.Fields[6].Axis = PivotAxisTypes.Row;
    pivotTable.Fields[3].Axis = PivotAxisTypes.Column;
    //Add data field
    IPivotField field = pivotTable.Fields[2];
    pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum);
    string fileName = "PivotTable.xlsx";
    //Saving the workbook as stream
    FileStream stream = new FileStream(fileName, FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("PivotTable.PivotData.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    IWorksheet pivotSheet = workbook.Worksheets[1];
    //Create Pivot cache with the given data range
    IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:H50"]);
    //Create "PivotTable1" with the cache at the specified range
    IPivotTable pivotTable = pivotSheet.PivotTables.Add("PivotTable1",
    pivotSheet["A1"], cache);
    //Add Pivot table fields (Row and Column fields)
    pivotTable.Fields[2].Axis = PivotAxisTypes.Row;
    pivotTable.Fields[6].Axis = PivotAxisTypes.Row;
    pivotTable.Fields[3].Axis = PivotAxisTypes.Column;
    //Add data field
    IPivotField field = pivotTable.Fields[2];
    pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum);
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
}

```

```

string fileName = "PivotTable.xlsx";
outputStream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies among Windows Phone,
//Android, and iOS platforms. Refer to the xlsio/xamarin section for
//respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
    "application/msexcel", outputStream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
    "application/msexcel", outputStream);
}
}

```

Editing and formatting a pivot table

A pivot table can be accessed from the **IPivotTables** interface that have the collection of pivot tables in the worksheet. The following code shows how to dynamically refresh the data in a pivot table. In prior:

- Create the pivot table using Excel GUI.
- Specify the named range to be the data source of the pivot table.
- Make sure that the "Refresh on Open" option of the pivot table is selected.
- Dynamically refresh the data in the named range.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("PivotTable.xlsx");
    IWorksheet pivotSheet = workbook.Worksheets[0];
    //Change the range values that the Pivot Tables range refers to
    workbook.Names["PivotRange"].RefersToRange = pivotSheet.Range["A1:D27"];
    workbook.SaveAs("PivotTable_DynamicRange.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("PivotTable.xlsx")
Dim pivotSheet As IWorksheet = workbook.Worksheets(0)
'Change the range values that the Pivot Tables range refers to
workbook.Names("PivotRange").RefersToRange = pivotSheet.Range("A1:D27")
workbook.SaveAs("PivotTable_DynamicRange.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("PivotTable.PivotTable.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream);
    IWorksheet pivotSheet = workbook.Worksheets[0];
    //Change the range values that the Pivot Tables range refers to
    workbook.Names["PivotRange"].RefersToRange = pivotSheet.Range["A1:D27"];
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "PivotTable_DynamicRange";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("PivotTable.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet pivotSheet = workbook.Worksheets[0];
    //Change the range values that the Pivot Tables range refers to
    workbook.Names["PivotRange"].RefersToRange = pivotSheet.Range["A1:D27"];
    string fileName = "PivotTable_DynamicRange.xlsx";
    //Saving the workbook as stream
    FileStream stream = new FileStream(fileName, FileMode.Create,
        FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection

```

```

Stream inputStream =
assembly.GetManifestResourceStream("PivotTable.PivotTable.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet pivotSheet = workbook.Worksheets[0];
//Change the range values that the Pivot Tables range refers to
workbook.Names["PivotRange"].RefersToRange = pivotSheet.Range["A1:D27"];
//Saving the workbook as stream
MemoryStream outputStream = new MemoryStream();
workbook.SaveAs(outputStream);
string fileName = "PivotTable_DynamicRange.xlsx";
outputStream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies among Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
"application/msexcel", outputStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
"application/msexcel", outputStream);
}
}

```

XlsIO supports 85 built-in styles of Excel 2007 used to create a table with rich formatting using the **PivotBuiltInStyles** property as follows. To learn more about various built-in styles supported, refer to the **PivotBuiltInStyles** enumeration in API section.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("PivotTable.xlsx");
IWorksheet worksheet = workbook.Worksheets[1];
IPivotTable pivotTable = worksheet.PivotTables[0];
//Set BuiltInStyle
pivotTable.BuiltInStyle = PivotBuiltInStyles.PivotStyleDark12;
workbook.SaveAs("PivotTable_Style.xlsx");
//No exception will be thrown if there are unsaved workbooks
excelEngine.ThrowNotSavedOnDestroy = false;
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("PivotTable.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(1)

```

```

Dim pivotTable As IPivotTable = sheet.PivotTables(0)
' Set BuiltInStyle
pivotTable.BuiltInStyle = PivotBuiltInStyles.PivotStyleDark12
workbook.SaveAs("PivotTable_Style.xlsx")
' No exception will be thrown if there are unsaved workbooks
excelEngine.ThrowNotSavedOnDestroy = False
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    // Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    // Gets input Excel document from embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("PivotTable.PivotTable.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream);
    IWorksheet worksheet = workbook.Worksheets[1];
    IPivotTable pivotTable = worksheet.PivotTables[0];
    // Set BuiltInStyle
    pivotTable.BuiltInStyle = PivotBuiltInStyles.PivotStyleDark12;
    // Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "PivotTable_Style";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx" });
    // Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    // Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("PivotTable.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[1];
    IPivotTable pivotTable = worksheet.PivotTables[0];
    // Set BuiltInStyle
    pivotTable.BuiltInStyle = PivotBuiltInStyles.PivotStyleDark12;
    string fileName = "PivotTable_Style.xlsx";
    // Saving the workbook as stream
    FileStream stream = new FileStream(fileName, FileMode.Create,
        FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("PivotTable.PivotTable.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[1];
    IPivotTable pivotTable = worksheet.PivotTables[0];
    //Set BuiltInStyle
    pivotTable.BuiltInStyle = PivotBuiltInStyles.PivotStyleDark12;
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
    string fileName = "PivotTable_Style.xlsx";
    outputStream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
            "application/msexcel", outputStream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
            "application/msexcel", outputStream);
    }
}

```

Refresh a pivot table

When you update the pivot table data source, refresh the pivot table manually to load the new data source into it. Essential XlsIO supports this refreshing of pivot table data source through **IsRefreshOnLoad** property of **PivotCacheImpl**.

Refer the following complete code snippets.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Updating a new value in the pivot data
    worksheet.Range["C2"].Value = "250";
}

```

```
//Accessing the pivot table
IPivotTable pivotTable = worksheet.PivotTables[0];
PivotTableImpl pivotTableImpl = pivotTable as PivotTableImpl;
//Refreshing pivot cache to update the pivot table
pivotTableImpl.Cache.IsRefreshOnLoad = true;
workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Updating a new value in the pivot data
worksheet.Range("C2").Value = "250"
'Accessing the pivot table
Dim pivotTable As IPivotTable = worksheet.PivotTables(0)
Dim pivotTableImpl As PivotTableImpl = CType(pivotTable, PivotTableImpl)
'Refreshing pivot cache to update the pivot table
pivotTableImpl.Cache.IsRefreshOnLoad = True
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Instantiates the file picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opening an existing workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(file);
IWorksheet worksheet = workbook.Worksheets[0];
//Updating a new value in the pivot data
worksheet.Range["C2"].Value = "250";
//Accessing the pivot table
IPivotTable pivotTable = worksheet.PivotTables[0];
PivotTableImpl pivotTableImpl = pivotTable as PivotTableImpl;
//Refreshing pivot cache to update the pivot table
pivotTableImpl.Cache.IsRefreshOnLoad = true;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
```

```
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Updating a new value in the pivot data
    worksheet.Range["C2"].Value = "250";
    //Accessing the pivot table
    IPivotTable pivotTable = worksheet.PivotTables[0];
    PivotTableImpl pivotTableImpl = pivotTable as PivotTableImpl;
    //Refreshing pivot cache to update the pivot table
    pivotTableImpl.Cache.IsRefreshOnLoad = true;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    // "App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("PivotTable.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Updating a new value in the pivot data
    worksheet.Range["C2"].Value = "250";
    //Accessing the pivot table
    IPivotTable pivotTable = worksheet.PivotTables[0];
    PivotTableImpl pivotTableImpl = pivotTable as PivotTableImpl;
    //Refreshing pivot cache to update the pivot table
    pivotTableImpl.Cache.IsRefreshOnLoad = true;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android, and iOS platforms. Refer to xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
```



```
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}
```

Expand or collapse rows in pivot table

Essential XlsIO allows you to expand and collapse the **PivotFieldItems** or simply the pivot table rows using **IsHiddenDetails** of **PivotItemOptions**.

Refer the following complete code snippets.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create pivot cache with the given data range
    IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:C13"]);
    //Create "PivotTable1" with the cache at the specified range
    IPivotTable pivotTable = worksheet.PivotTables.Add("PivotTable1",
    worksheet["E1"], cache);
    //Add pivot table fields (Row and Column fields)
    pivotTable.Fields[0].Axis = PivotAxisTypes.Row;
    pivotTable.Fields[1].Axis = PivotAxisTypes.Row;
    //Add data field
    IPivotField field = pivotTable.Fields[2];
    pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum);
    //Initialize PivotItemOptions
    PivotItemOptions options = new PivotItemOptions();
    options.IsHiddenDetails = false;
    //Collapsing the first and second items of the first pivot field using
    PivotItemOptions
    (pivotTable.Fields[0] as PivotFieldImpl).AddItemOption(0, options);
    (pivotTable.Fields[0] as PivotFieldImpl).AddItemOption(1, options);
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Create pivot cache with the given data range
Dim cache As IPivotCache = workbook.PivotCaches.Add(worksheet("A1:C13"))
'Create "PivotTable1" with the cache at the specified range
```

```

Dim pivotTable As IPivotTable = worksheet.PivotTables.Add("PivotTable1",
worksheet("E1"), cache)
'Add pivot table fields (Row and Column fields)
pivotTable.Fields(0).Axis = PivotAxisTypes.Row
pivotTable.Fields(1).Axis = PivotAxisTypes.Row
'Add data field
Dim field As IPivotField = pivotTable.Fields(2)
pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum)
'Initialize PivotItemOptions
Dim options As PivotItemOptions = New PivotItemOptions
options.IsHiddenDetails = False
'Collapsing the first and second items of the first pivot field using
PivotItemOptions
CType(pivotTable.Fields(0), PivotFieldImpl).AddItemOption(0, options)
CType(pivotTable.Fields(0), PivotFieldImpl).AddItemOption(1, options)
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Instantiates the file picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opening the existing workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create pivot cache with the given data range
    IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:C13"]);
    //Create "PivotTable1" with the cache at the specified range
    IPivotTable pivotTable = worksheet.PivotTables.Add("PivotTable1",
worksheet["E1"], cache);
    //Add pivot table fields (Row and Column fields)
    pivotTable.Fields[0].Axis = PivotAxisTypes.Row;
    pivotTable.Fields[1].Axis = PivotAxisTypes.Row;
    //Add data field
    IPivotField field = pivotTable.Fields[2];
    pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum);
    //Initialize PivotItemOptions
    PivotItemOptions options = new PivotItemOptions();
    options.IsHiddenDetails = false;
    //Collapsing the first and second items of the first pivot field using
    PivotItemOptions
    (pivotTable.Fields[0] as PivotFieldImpl).AddItemOption(0, options);
    (pivotTable.Fields[0] as PivotFieldImpl).AddItemOption(1, options);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
}

```

```

savePicker.SuggestedFileName = "Output";
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create pivot cache with the given data range
    IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:C13"]);
    //Create "PivotTable1" with the cache at the specified range
    IPivotTable pivotTable = worksheet.PivotTables.Add("PivotTable1",
    worksheet["E1"], cache);
    //Add pivot table fields (Row and Column fields)
    pivotTable.Fields[0].Axis = PivotAxisTypes.Row;
    pivotTable.Fields[1].Axis = PivotAxisTypes.Column;
    //Add data field
    IPivotField field = pivotTable.Fields[2];
    pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum);
    //Initialize PivotItemOptions
    PivotItemOptions options = new PivotItemOptions();
    options.IsHiddenDetails = false;
    //Collapsing the first and second items of the first pivot field using
    PivotItemOptions
    (pivotTable.Fields[0] as PivotFieldImpl).AddItemOption(0, options);
    (pivotTable.Fields[0] as PivotFieldImpl).AddItemOption(1, options);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    // "App" is the class of portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("PivotTable.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create pivot cache with the given data range
    IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:C13"]);
}

```

```

//Create "PivotTable1" with the cache at the specified range
IPivotTable pivotTable = worksheet.PivotTables.Add("PivotTable1",
worksheet["E1"], cache);
//Add pivot table fields (Row and Column fields)
pivotTable.Fields[0].Axis = PivotAxisTypes.Row;
pivotTable.Fields[1].Axis = PivotAxisTypes.Row;
//Add data field
IPivotField field = pivotTable.Fields[2];
pivotTable.DataFields.Add(field, "Sum", PivotSubtotalTypes.Sum);
//Initialize PivotItemOptions
PivotItemOptions options = new PivotItemOptions();
options.IsHiddenDetails = false;
//Collapsing the first and second items of the first pivot field using
PivotItemOptions
(pivotTable.Fields[0] as PivotFieldImpl).AddItemOption(0, options);
(pivotTable.Fields[0] as PivotFieldImpl).AddItemOption(1, options);
//Saving the workbooks as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Applying pivot table filters

The filtered data of a pivot table displays only the subset of data that meets the specified criteria. This can be achieved in XlsIO using the **IPivotFilters** interface.

Applying page filters

The page field filter or report filter can filter the pivot table based on the page field items. The following code snippet illustrates how to apply multiple filters to the page field items.

C#

```

//Set field axis to page
pivotTable.Fields[4].Axis = PivotAxisTypes.Page;
//Apply page field filter
IPivotField pageField = pivotTable.Fields[4];
//Select multiple items in page field to filter
pageField.Items[1].Visible = false;
pageField.Items[2].Visible = false;

```

VB.NET

```
'Set field axis to page
pivotTable.Fields(4).Axis = PivotAxisTypes.Page
'Apply page field filter
Dim pageField As IPivotField = pivotTable.Fields(4)
'Select multiple items in page field to filter
pageField.Items(1).Visible = False
pageField.Items(2).Visible = False
```

UWP

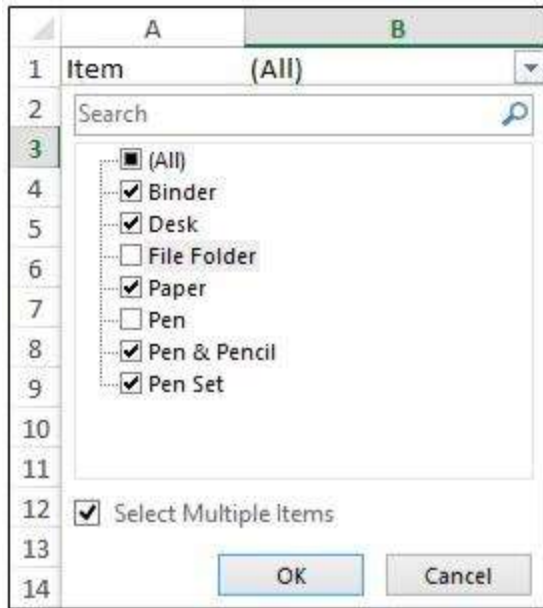
```
//Set field axis to page
pivotTable.Fields[4].Axis = PivotAxisTypes.Page;
//Apply page field filter
IPivotField pageField = pivotTable.Fields[4];
//Select multiple items in page field to filter
pageField.Items[1].Visible = false;
pageField.Items[2].Visible = false;
```

ASP.NET CORE

```
//Set field axis to page
pivotTable.Fields[4].Axis = PivotAxisTypes.Page;
//Apply page field filter
IPivotField pageField = pivotTable.Fields[4];
//Select multiple items in page field to filter
pageField.Items[1].Visible = false;
pageField.Items[2].Visible = false;
```

XAMARIN

```
//Set field axis to page
pivotTable.Fields[4].Axis = PivotAxisTypes.Page;
//Apply page field filter
IPivotField pageField = pivotTable.Fields[4];
//Select multiple items in page field to filter
pageField.Items[1].Visible = false;
pageField.Items[2].Visible = false;
```



Applying row or column filters

The row and column field filters can filter the pivot table based on labels, values, and items of the fields. The following code example illustrates how to apply these filters to a pivot table.

Label Filter

C#

```
//Apply row field filter
IPivotField rowField = pivotTable.Fields[2];
//Applying Label based row field filter
rowField.PivotFilters.Add(PivotFilterType.CaptionEqual, null, "Central",
null);
```

VB.NET

```
'Apply row field filter
Dim rowField As IPivotField = pivotTable.Fields(2)
'Applying Label based row field filter
rowField.PivotFilters.Add(PivotFilterType.CaptionEqual, Nothing, "Central",
Nothing)
```

UWP

```
//Apply row field filter
IPivotField rowField = pivotTable.Fields[2];
//Applying Label based row field filter
rowField.PivotFilters.Add(PivotFilterType.CaptionEqual, null, "Central",
null);
```

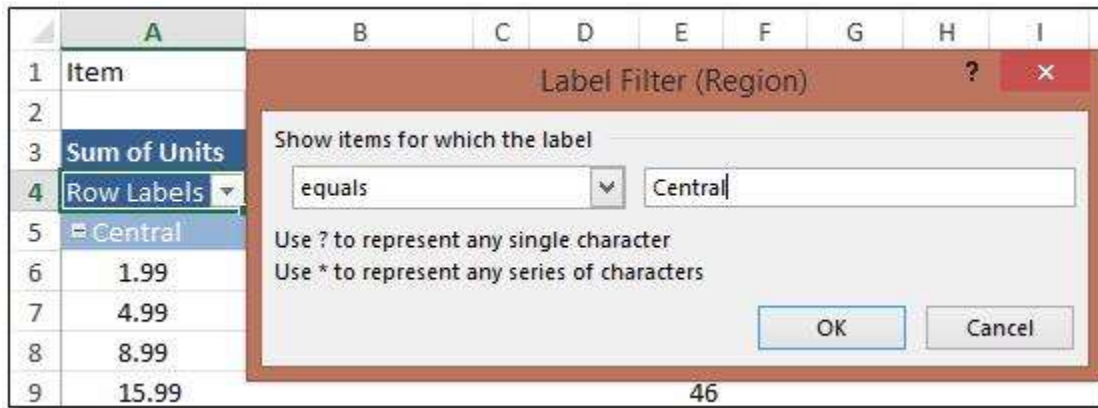
ASP.NET CORE

```
//Apply row field filter
IPivotField rowField = pivotTable.Fields[2];
```

```
//Applying Label based row field filter
rowField.PivotFilters.Add(PivotFilterType.CaptionEqual, null, "Central",
null);
```

XAMARIN

```
//Apply row field filter
IPivotField rowField = pivotTable.Fields[2];
//Applying Label based row field filter
rowField.PivotFilters.Add(PivotFilterType.CaptionEqual, null, "Central",
null);
```



Value Filter

C#

```
IPivotField field = pivotTable.Fields[2];
//Apply value filter
field.PivotFilters.Add(PivotFilterType.ValueLessThan, field, "1341", null);
```

VB.NET

```
'Apply row field filter
Dim field As IPivotField = pivotTable.Fields(2)
'Applying value filter
field.PivotFilters.Add(PivotFilterType.ValueLessThan, field, "1341",
Nothing)
```

UWP

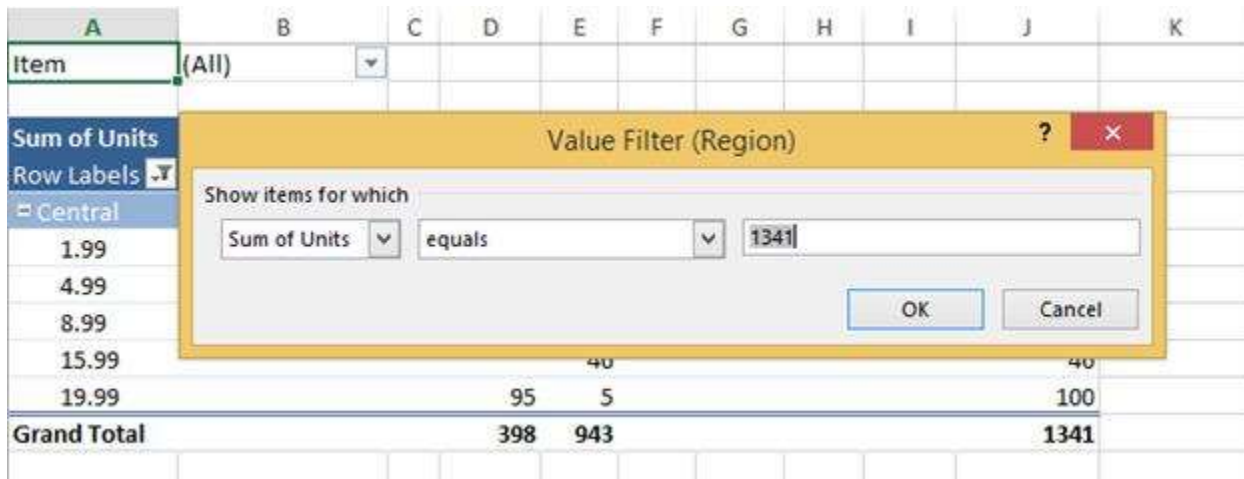
```
IPivotField field = pivotTable.Fields[2];
//Apply value filter
field.PivotFilters.Add(PivotFilterType.ValueLessThan, field, "1341", null);
```

ASP.NET CORE

```
IPivotField field = pivotTable.Fields[2];
//Apply value filter
field.PivotFilters.Add(PivotFilterType.ValueLessThan, field, "1341", null);
```

XAMARIN

```
IPivotField field = pivotTable.Fields[2];
//Apply value filter
field.PivotFilters.Add(PivotFilterType.ValueLessThan, field, "1341", null);
```



Item	(All)									
Sum of Units										
Row Labels										
Central										
1.99										
4.99										
8.99										
15.99										
19.99										
Grand Total										

Multiple filter**C#**

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("PivotData.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    IWorksheet pivotSheet = workbook.Worksheets[1];
    IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:H50"]);
    IPivotTable pivotTable = pivotSheet.PivotTables.Add("PivotTable1",
        pivotSheet["A1"], cache);
    pivotTable.Fields[4].Axis = PivotAxisTypes.Page;
    pivotTable.Fields[2].Axis = PivotAxisTypes.Row;
    pivotTable.Fields[6].Axis = PivotAxisTypes.Row;
    pivotTable.Fields[3].Axis = PivotAxisTypes.Column;
    IPivotField dataField = pivotSheet.PivotTables[0].Fields[5];
    pivotTable.DataFields.Add(dataField, "Sum of Units",
        PivotSubtotalTypes.Sum);
    //Apply page filter
    pivotTable.Fields[4].Axis = PivotAxisTypes.Page;
    IPivotField pageField = pivotTable.Fields[4];
    pageField.Items[1].Visible = false;
    pageField.Items[2].Visible = false;
    //Apply label filter
    IPivotField rowField = pivotTable.Fields[2];
    rowField.PivotFilters.Add(PivotFilterType.CaptionEqual, null, "East", null);
    //Apply item filter
    IPivotField colField = pivotTable.Fields[3];
    colField.Items[0].Visible = false;
    colField.Items[1].Visible = false;
    //Apply value filter
```



```

IPivotField field = pivotTable.Fields[2];
field.PivotFilters.Add(PivotFilterType.ValueLessThan, field, "1341", null);
pivotTable.BuiltInStyle = PivotBuiltInStyles.PivotStyleMedium2;
pivotSheet.Activate();
workbook.SaveAs("PivotTable.xlsx");
//No exception will be thrown if there are unsaved workbooks.
excelEngine.ThrowNotSavedOnDestroy = false;
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("PivotData.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim pivotSheet As IWorksheet = workbook.Worksheets(1)
Dim cache As IPivotCache = workbook.PivotCaches.Add(worksheet("A1:H50"))
Dim pivotTable As IPivotTable = pivotSheet.PivotTables.Add("PivotTable1",
pivotSheet("A1"), cache)
pivotTable.Fields(4).Axis = PivotAxisTypes.Page
pivotTable.Fields(2).Axis = PivotAxisTypes.Row
pivotTable.Fields(6).Axis = PivotAxisTypes.Row
pivotTable.Fields(3).Axis = PivotAxisTypes.Column
Dim dataField As IPivotField = pivotSheet.PivotTables(0).Fields(5)
pivotTable.DataFields.Add(dataField, "Sum of Units", PivotSubtotalTypes.Sum)
'Applying page filter
pivotTable.Fields(4).Axis = PivotAxisTypes.Page
Dim pageField As IPivotField = pivotTable.Fields(4)
pageField.Items(1).Visible = False
pageField.Items(2).Visible = False
'Apply label filter
Dim rowField As IPivotField = pivotTable.Fields(2)
rowField.PivotFilters.Add(PivotFilterType.CaptionEqual, Nothing, "East",
Nothing)
'Apply item filter
Dim colField As IPivotField = pivotTable.Fields(3)
colField.Items(0).Visible = False
colField.Items(1).Visible = False
'Applying value filter
Dim field As IPivotField = pivotTable.Fields(2)
field.PivotFilters.Add(PivotFilterType.ValueLessThan, field, "1341",
Nothing)
pivotTable.BuiltInStyle = PivotBuiltInStyles.PivotStyleMedium2
pivotSheet.Activate()
workbook.SaveAs("PivotTable.xlsx")
'No exception will be thrown if there are unsaved workbooks
excelEngine.ThrowNotSavedOnDestroy = False
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;

```

```

//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from embedded resource collection
Stream inputStream =
assembly.GetManifestResourceStream("PivotTable.PivotData.xlsx");
IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream);
IWorksheet worksheet = workbook.Worksheets[0];
IWorksheet pivotSheet = workbook.Worksheets[1];
IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:H50"]);
IPivotTable pivotTable = pivotSheet.PivotTables.Add("PivotTable1",
pivotSheet["A1"], cache);
pivotTable.Fields[4].Axis = PivotAxisTypes.Page;
pivotTable.Fields[2].Axis = PivotAxisTypes.Row;
pivotTable.Fields[6].Axis = PivotAxisTypes.Row;
pivotTable.Fields[3].Axis = PivotAxisTypes.Column;
IPivotField dataField = pivotSheet.PivotTables[0].Fields[5];
pivotTable.DataFields.Add(dataField, "Sum of Units",
PivotSubtotalTypes.Sum);
//Apply page filter
pivotTable.Fields[4].Axis = PivotAxisTypes.Page;
IPivotField pageField = pivotTable.Fields[4];
pageField.Items[1].Visible = false;
pageField.Items[2].Visible = false;
//Apply label filter
IPivotField rowField = pivotTable.Fields[2];
rowField.PivotFilters.Add(PivotFilterType.CaptionEqual, null, "East", null);
//Apply item filter
IPivotField colField = pivotTable.Fields[3];
colField.Items[0].Visible = false;
colField.Items[1].Visible = false;
//Apply value filter
IPivotField field = pivotTable.Fields[2];
field.PivotFilters.Add(PivotFilterType.ValueLessThan, field, "1341", null);
pivotTable.BuiltInStyle = PivotBuiltInStyles.PivotStyleMedium2;
pivotSheet.Activate();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "PivotTable";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
FileStream fileStream = new FileStream("PivotData.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream);
IWorksheet worksheet = workbook.Worksheets[0];

```

```

IWorksheet pivotSheet = workbook.Worksheets[1];
IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:H50"]);
IPivotTable pivotTable = pivotSheet.PivotTables.Add("PivotTable1",
pivotSheet["A1"], cache);
pivotTable.Fields[4].Axis = PivotAxisTypes.Page;
pivotTable.Fields[2].Axis = PivotAxisTypes.Row;
pivotTable.Fields[6].Axis = PivotAxisTypes.Row;
pivotTable.Fields[3].Axis = PivotAxisTypes.Column;
IPivotField dataField = pivotSheet.PivotTables[0].Fields[5];
pivotTable.DataFields.Add(dataField, "Sum of Units",
PivotSubtotalTypes.Sum);
//Apply page filter
pivotTable.Fields[4].Axis = PivotAxisTypes.Page;
IPivotField pageField = pivotTable.Fields[4];
pageField.Items[1].Visible = false;
pageField.Items[2].Visible = false;
//Apply label filter
IPivotField rowField = pivotTable.Fields[2];
rowField.PivotFilters.Add(PivotFilterType.CaptionEqual, null, "East", null);
//Apply item filter
IPivotField colField = pivotTable.Fields[3];
colField.Items[0].Visible = false;
colField.Items[1].Visible = false;
//Apply value filter
IPivotField field = pivotTable.Fields[2];
field.PivotFilters.Add(PivotFilterType.ValueLessThan, field, "1341", null);
pivotTable.BuiltInStyle = PivotBuiltInStyles.PivotStyleMedium2;
pivotSheet.Activate();
string fileName = "PivotTable.xlsx";
//Saving the workbook as stream
FileStream stream = new FileStream(fileName, FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from embedded resource collection
Stream inputStream =
assembly.GetManifestResourceStream("PivotTable.PivotData.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet worksheet = workbook.Worksheets[0];
IWorksheet pivotSheet = workbook.Worksheets[1];
IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:H50"]);
IPivotTable pivotTable = pivotSheet.PivotTables.Add("PivotTable1",
pivotSheet["A1"], cache);
pivotTable.Fields[4].Axis = PivotAxisTypes.Page;
pivotTable.Fields[2].Axis = PivotAxisTypes.Row;
pivotTable.Fields[6].Axis = PivotAxisTypes.Row;

```

```

pivotTable.Fields[3].Axis = PivotAxisTypes.Column;
IPivotField dataField = pivotSheet.PivotTables[0].Fields[5];
pivotTable.DataFields.Add(dataField, "Sum of Units",
PivotSubtotalTypes.Sum);
//Apply page filter
pivotTable.Fields[4].Axis = PivotAxisTypes.Page;
IPivotField pageField = pivotTable.Fields[4];
pageField.Items[1].Visible = false;
pageField.Items[2].Visible = false;
//Apply label filter
IPivotField rowField = pivotTable.Fields[2];
rowField.PivotFilters.Add(PivotFilterType.CaptionEqual, null, "East", null);
//Apply item filter
IPivotField colField = pivotTable.Fields[3];
colField.Items[0].Visible = false;
colField.Items[1].Visible = false;
//Apply value filter
IPivotField field = pivotTable.Fields[2];
field.PivotFilters.Add(PivotFilterType.ValueLessThan, field, "1341", null);
pivotTable.BuiltInStyle = PivotBuiltInStyles.PivotStyleMedium2;
pivotSheet.Activate();
//Saving the workbook as stream
MemoryStream outputStream = new MemoryStream();
workbook.SaveAs(outputStream);
string fileName = "PivotTable.xlsx";
outputStream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies among Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
"application/msexcel", outputStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
"application/msexcel", outputStream);
}
}

```

Applying pivot table settings

Excel provides various options through the PivotTableOptions dialog box to customize the appearance of the pivot table.

XlsIO supports these pivot table options using the IPivotTableOptions interface to control various settings for the existing pivot table. To learn more about various pivot table options, refer to the **IPivotTableOptions** in API section. The following code illustrates how to access the PivotTableOptions object.

C#

```

//Enable ColumnHeaderCaption
IPivotTable pivotTable = worksheet.PivotTables[0];

```

```
IPivotTableOptions options = pivotTable.Options;
```

VB.NET

```
'Enable ColumnHeaderCaption  
Dim pivotTable As IPivotTable = worksheet.PivotTables(0)  
Dim options As IPivotTableOptions = pivotTable.Options
```

UWP

```
//Enable ColumnHeaderCaption  
IPivotTable pivotTable = worksheet.PivotTables[0];  
IPivotTableOptions options = pivotTable.Options;
```

ASP.NET CORE

```
//Enable ColumnHeaderCaption  
IPivotTable pivotTable = worksheet.PivotTables[0];  
IPivotTableOptions options = pivotTable.Options;
```

XAMARIN

```
//Enable ColumnHeaderCaption  
IPivotTable pivotTable = worksheet.PivotTables[0];  
IPivotTableOptions options = pivotTable.Options;
```

Show or hide the field list

To show or hide the pivot table field list pane, use the ShowFieldList property.

C#

```
//Enable ShowFieldList  
options.ShowFieldList = false;
```

VB.NET

```
'Enable ShowFieldList  
options.ShowFieldList = False
```

UWP

```
//Enable ShowFieldList  
options.ShowFieldList = false;
```

ASP.NET CORE

```
//Enable ShowFieldList  
options.ShowFieldList = false;
```

XAMARIN

```
//Enable ShowFieldList
```

```
options.ShowFieldList = false;
```

Header caption

The **RowHeaderCaption** and **ColumnHeaderCaption** properties allows to edit the respective pivot table headers. The header caption can be enabled or disabled using the **DisplayFieldCaption** property.

C#

```
//Enable header captions
options.RowHeaderCaption = "Payment Dates";
options.ColumnHeaderCaption = "Payments";
```

VB.NET

```
'Enable header captions
options.RowHeaderCaption = "Payment Dates"
options.ColumnHeaderCaption = "Payments"
```

UWP

```
//Enable header captions
options.RowHeaderCaption = "Payment Dates";
options.ColumnHeaderCaption = "Payments";
```

ASP.NET CORE

```
//Enable header captions
options.RowHeaderCaption = "Payment Dates";
options.ColumnHeaderCaption = "Payments";
```

XAMARIN

```
//Enable header captions
options.RowHeaderCaption = "Payment Dates";
options.ColumnHeaderCaption = "Payments";
```

Grand total

XlsIO provides an equivalent API to perform grand totals with the properties as follows.

C#

```
//Enable GrandTotals
pivotTable.ColumnGrand = false;
pivotTable.RowGrand = true;
```

VB.NET

```
'Enable GrandTotals
pivotTable.ColumnGrand = False
pivotTable.RowGrand = False
```

UWP

```
//Enable GrandTotals  
pivotTable.ColumnGrand = false;  
pivotTable.RowGrand = true;
```

ASP.NET CORE

```
//Enable GrandTotals  
pivotTable.ColumnGrand = false;  
pivotTable.RowGrand = true;
```

XAMARIN

```
//Enable GrandTotals  
pivotTable.ColumnGrand = false;  
pivotTable.RowGrand = true;
```

Show or hide collapse button

You can also show or hide the **Collapse** button that appears in the fields of the pivot table, when more than one item exists in a field. The following code example illustrates how to do this.

C#

```
//Enable ShowDrillIndicators  
pivotTable.ShowDrillIndicators = true;
```

VB.NET

```
'Enable ShowDrillIndicators  
pivotTable.ShowDrillIndicators = True
```

UWP

```
//Enable ShowDrillIndicators  
pivotTable.ShowDrillIndicators = true;
```

ASP.NET CORE

```
//Enable ShowDrillIndicators  
pivotTable.ShowDrillIndicators = true;
```

XAMARIN

```
//Enable ShowDrillIndicators  
pivotTable.ShowDrillIndicators = true;
```

Display field caption and filter option

The filter buttons and field names in the pivot table can be shown or hidden, as in the following code.

C#

```
//Enable DisplayFieldCaption  
pivotTable.DisplayFieldCaptions = true;
```

VB.NET

```
'Enable DisplayFieldCaption  
pivotTable.DisplayFieldCaptions = True
```

UWP

```
//Enable DisplayFieldCaption  
pivotTable.DisplayFieldCaptions = true;
```

ASP.NET CORE

```
//Enable DisplayFieldCaption  
pivotTable.DisplayFieldCaptions = true;
```

XAMARIN

```
//Enable DisplayFieldCaption  
pivotTable.DisplayFieldCaptions = true;
```

Repeating row label on each page

You can set the row label on each page while printing, and the header can be viewed on each page.

C#

```
//Enable RepeatItemsOnEachPrintedPage  
pivotTable.RepeatItemsOnEachPrintedPage = true;
```

VB.NET

```
'Enable RepeatItemsOnEachPrintedPage  
pivotTable.RepeatItemsOnEachPrintedPage = True
```

UWP

```
//Enable RepeatItemsOnEachPrintedPage  
pivotTable.RepeatItemsOnEachPrintedPage = true;
```

ASP.NET CORE

```
//Enable RepeatItemsOnEachPrintedPage  
pivotTable.RepeatItemsOnEachPrintedPage = true;
```

XAMARIN

```
//Enable RepeatItemsOnEachPrintedPage  
pivotTable.RepeatItemsOnEachPrintedPage = true;
```


Repeat Labels

You can repeat labels for row or column fields when the [pivot table layout](#) is set to tabular or outline layout forms.

Specific Pivot Field

The following code illustrates how to set the repeat labels option to a specific pivot field.

C#

```
//Set repeat labels option to a specific pivot field  
pivotTable.Fields[0].RepeatLabels = true;
```

VB.NET

```
'Set repeat labels option to a specific pivot field  
pivotTable.Fields(0).RepeatLabels = True
```

UWP

```
//Set repeat labels option to a specific pivot field  
pivotTable.Fields[0].RepeatLabels = true;
```

ASP.NET CORE

```
//Set repeat labels option to a specific pivot field  
pivotTable.Fields[0].RepeatLabels = true;
```

XAMARIN

```
//Set repeat labels option to a specific pivot field  
pivotTable.Fields[0].RepeatLabels = true;
```

All Pivot Fields

The following code illustrates how to set the repeat labels option to all the pivot fields.

C#

```
//Set repeat labels option to all the pivot fields  
pivotTable.Options.RepeatAllLabels(true);
```

VB.NET

```
'Set repeat labels option to all the pivot fields  
pivotTable.Options.RepeatAllLabels(True)
```

UWP

```
//Set repeat labels option to all the pivot fields  
pivotTable.Options.RepeatAllLabels(true);
```

ASP.NET CORE

```
//Set repeat labels option to all the pivot fields
pivotTable.Options.RepeatAllLabels(true);
```

XAMARIN

```
//Set repeat labels option to all the pivot fields
pivotTable.Options.RepeatAllLabels(true);
```

Sort by value in Pivot Table

Pivot field AutoSort allows you to sort the pivot row or column fields based on the data field values. You can perform the sorting in following direction:

- Top to Bottom
- Left to Right

Sort a Pivot Table Field Top to Bottom

Top to Bottom sorting can sort the pivot table column field values based on the sort type. To apply Top to Bottom sorting in pivot table, you should apply the sorting in pivot row field by AutoSort method. The following code example illustrates how to apply Top to Bottom sorting to a pivot table.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("PivotTable.xlsx");
    IWorksheet sheet = workbook.Worksheets[1];
    IPivotTable pivotTable = sheet.PivotTables[0];
    // Pivot Top to Bottom sorting.
    IPivotField rowField = pivotTable.RowFields[0];
    rowField.AutoSort(PivotFieldSortType.Ascending, 1);
    workbook.SaveAs("PivotFieldAutoSort.xlsx");
    excelEngine.ThrowNotSavedOnDestroy = false;
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("PivotTable.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(1)
Dim pivotTable As IPivotTable = sheet.PivotTables(0)
' Pivot Top to Bottom sorting.
Dim rowField As IPivotField = pivotTable.RowFields(0)
rowField.AutoSort(PivotFieldSortType.Ascending, 1)
workbook.SaveAs("PivotTableCalculate.xlsx")
excelEngine.ThrowNotSavedOnDestroy = False
End Using
```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("PivotTable.PivotTable.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream);
    IWorksheet sheet = workbook.Worksheets[1];
    IPivotTable pivotTable = sheet.PivotTables[0];
    // Pivot Top to Bottom sorting.
    IPivotField rowField = pivotTable.RowFields[0];
    rowField.AutoSort(PivotFieldSortType.Ascending, 1);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "PivotFieldAutoSort";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("PivotTable.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet sheet = workbook.Worksheets[1];
    IPivotTable pivotTable = sheet.PivotTables[0];
    // Pivot Top to Bottom sorting.
    IPivotField rowField = pivotTable.RowFields[0];
    rowField.AutoSort(PivotFieldSortType.Ascending, 1);
    string fileName = "PivotFieldAutoSort.xlsx";
    //Saving the workbook as stream
    FileStream stream = new FileStream(fileName, FileMode.Create,
        FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly

```

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from embedded resource collection
Stream inputStream =
assembly.GetManifestResourceStream("PivotTable.PivotTable.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet sheet = workbook.Worksheets[1];
IPivotTable pivotTable = sheet.PivotTables[0];
// Pivot Top to Bottom sorting.
IPivotField rowField = pivotTable.RowFields[0];
rowField.AutoSort(PivotFieldSortType.Ascending, 1);
//Saving the workbook as stream
MemoryStream outputStream = new MemoryStream();
workbook.SaveAs(outputStream);
string fileName = "PivotFieldAutoSort.xlsx";
outputStream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies among Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
"application/msexcel", outputStream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
"application/msexcel", outputStream);
}
}

```

Sort a Pivot Table Field Left to Right

Left to Right sorting can sort the pivot table row field values based on the sort type. To apply Left to Right sorting in pivot table, you should apply the sorting in pivot column field by AutoSort method. The following code example illustrates how to apply Left to Right sorting to a pivot table.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("PivotTable.xlsx");
IWorksheet sheet = workbook.Worksheets[1];
IPivotTable pivotTable = sheet.PivotTables[0];
// Pivot table Left to Right sorting.
IPivotField columnField = pivotTable.ColumnFields[0];
columnField.AutoSort(PivotFieldSortType.Ascending, 1);
workbook.SaveAs("PivotFieldAutoSort.xlsx");
excelEngine.ThrowNotSavedOnDestroy = false;
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("PivotTable.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(1)
Dim pivotTable As IPivotTable = sheet.PivotTables(0)
' Pivot table Left to Right sorting.
Dim rowField As columnField = pivotTable.ColumnFields(0)
columnField.AutoSort(PivotFieldSortType.Ascending, 1)
workbook.SaveAs("PivotTableCalculate.xlsx")
excelEngine.ThrowNotSavedOnDestroy = False
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("PivotTable.PivotTable.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream);
    IWorksheet sheet = workbook.Worksheets[1];
    IPivotTable pivotTable = sheet.PivotTables[0];
    // Pivot table Left to Right sorting.
    IPivotField columnField = pivotTable.ColumnFields[0];
    columnField.AutoSort(PivotFieldSortType.Ascending, 1);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "PivotFieldAutoSort";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("PivotTable.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet sheet = workbook.Worksheets[1];
    IPivotTable pivotTable = sheet.PivotTables[0];
    // Pivot table Left to Right sorting.
    IPivotField columnField = pivotTable.ColumnFields[0];
    columnField.AutoSort(PivotFieldSortType.Ascending, 1);
}

```

```

string fileName = "PivotFieldAutoSort.xlsx";
//Saving the workbook as stream
FileStream stream = new FileStream(fileName, FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("PivotTable.PivotTable.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[1];
    IPivotTable pivotTable = sheet.PivotTables[0];
    // Pivot table Left to Right sorting.
    IPivotField columnField = pivotTable.ColumnFields[0];
    columnField.AutoSort(PivotFieldSortType.Ascending, 1);
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
    string fileName = "PivotFieldAutoSort.xlsx";
    outputStream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
        "application/msexcel", outputStream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
        "application/msexcel", outputStream);
    }
}

```

Note: PivotCacheImpl.IsRefreshOnLoad property is set as true when applying AutoSort to pivot fields.

Other pivot table operations

Adding calculated field in the existing pivot table

Calculated field is a special type of database field that perform calculations by using the contents of other fields in the pivot table with the given formula. The formula can contain operators and expressions as in other worksheet formulas. You can use constants and refer to the data from the PivotTable.

You can read and create the calculated fields in the existing pivot table. The following are the Excel restrictions when using the formula:

- Formula cannot contain cell references or defined names.
- Formula cannot contain Worksheet functions that require cell references.
- Formula cannot use array functions.

The calculated field in XlsIO can be achieved using the following code sample.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("PivotTable.xlsx");
    IWorksheet sheet = workbook.Worksheets[1];
    IPivotTable pivotTable = sheet.PivotTables[0];
    //Add calculated field to the first pivot table
    IPivotField field = pivotTable.CalculatedFields.Add("Percent",
        "Sales/Total*100");
    workbook.SaveAs("PivotTableCalculate.xlsx");
    excelEngine.ThrowNotSavedOnDestroy = false;
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("PivotTable.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(1)
Dim pivotTable As IPivotTable = sheet.PivotTables(0)
' Add calculated field to the first pivot table
Dim field As IPivotField = pivotTable.CalculatedFields.Add("Percent",
    "Sales/Total*100")
workbook.SaveAs("PivotTableCalculate.xlsx")
excelEngine.ThrowNotSavedOnDestroy = False
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("PivotTable.PivotTable.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream);
    IWorksheet sheet = workbook.Worksheets[1];
    IPivotTable pivotTable = sheet.PivotTables[0];
    //Add calculated field to the first pivot table
}
```

```

IPivotField field = pivotTable.CalculatedFields.Add("Percent",
"Sales/Total*100");
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "PivotTableCalculate";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("PivotTable.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet sheet = workbook.Worksheets[1];
    IPivotTable pivotTable = sheet.PivotTables[0];
    //Add calculated field to the first pivot table
    IPivotField field = pivotTable.CalculatedFields.Add("Percent",
    "Sales/Total*100");
    string fileName = "PivotTableCalculate.xlsx";
    //Saving the workbook as stream
    FileStream stream = new FileStream(fileName, FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("PivotTable.PivotTable.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[1];
    IPivotTable pivotTable = sheet.PivotTables[0];
    //Add calculated field to the first pivot table
    IPivotField field = pivotTable.CalculatedFields.Add("Percent",
    "Sales/Total*100");
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
}

```



```

string fileName = "PivotTableCalculate.xlsx";
outputStream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies among Windows Phone,
//Android, and iOS platforms. Refer to the xlsio/xamarin section for
//respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
    "application/msexcel", outputStream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
    "application/msexcel", outputStream);
}
}

```

The formula can also be set to the IPivotField property as follows.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("PivotTable.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    IPivotTable pivotTable = sheet.PivotTables[0];
    //Add calculated field to the first pivot table
    IPivotField field = pivotTable.CalculatedFields.Add("Percent",
    "Sales/Total*100");
    //Set Field Formula
    field.Formula = "Sales/Total*200";
    workbook.SaveAs("PivotTable.xlsx");
    excelEngine.ThrowNotSavedOnDestroy = false;
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("PivotTable.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim pivotTable As IPivotTable = sheet.PivotTables(0)
Dim field As IPivotField = pivotTable.CalculatedFields.Add("Percent",
"Sales/Total*100")
'Set Field Formula
field.Formula = "Sales/Total*200"
workbook.SaveAs("PivotTable.xlsx")
excelEngine.ThrowNotSavedOnDestroy = False
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("PivotTable.PivotTable.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream);
    IWorksheet sheet = workbook.Worksheets[1];
    IPivotTable pivotTable = sheet.PivotTables[0];
    //Add calculated field to the first pivot table
    IPivotField field = pivotTable.CalculatedFields.Add("Percent",
        "Sales/Total*100");
    //Set Field Formula
    field.Formula = "Sales/Total*200";
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "PivotTableCalculate";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("PivotTable.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet sheet = workbook.Worksheets[1];
    IPivotTable pivotTable = sheet.PivotTables[0];
    //Add calculated field to the first pivot table
    IPivotField field = pivotTable.CalculatedFields.Add("Percent",
        "Sales/Total*100");
    //Set Field Formula
    field.Formula = "Sales/Total*200";
    string fileName = "PivotTableCalculate.xlsx";
    //Saving the workbook as stream
    FileStream stream = new FileStream(fileName, FileMode.Create,
        FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("PivotTable.PivotTable.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[1];
    IPivotTable pivotTable = sheet.PivotTables[0];
    //Add calculated field to the first pivot table
    IPivotField field = pivotTable.CalculatedFields.Add("Percent",
        "Sales/Total*100");
    //Set Field Formula
    field.Formula = "Sales/Total*200";
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
    string fileName = "PivotTableCalculate.xlsx";
    outputStream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
            "application/msexcel", outputStream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
            "application/msexcel", outputStream);
    }
}

```

Layout the pivot table like Excel

A pivot table can be created similar to the Excel layout.

The following code example illustrates how to enable Essential XlsIO to layout the pivot table like Excel.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("PivotData.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    IWorksheet pivotSheet = workbook.Worksheets[1];
    IPivotCache cache = workbook.PivotCaches.Add(worksheet["A1:H50"]);
}

```

```

IPivotTable pivotTable = pivotSheet.PivotTables.Add("PivotTable1",
pivotSheet["A1"], cache);
pivotTable.Fields[4].Axis = PivotAxisTypes.Page;
pivotTable.Fields[2].Axis = PivotAxisTypes.Row;
pivotTable.Fields[6].Axis = PivotAxisTypes.Row;
pivotTable.Fields[3].Axis = PivotAxisTypes.Column;
IPivotField dataField = pivotSheet.PivotTables[0].Fields[5];
pivotTable.DataFields.Add(dataField, "Sum of Units",
PivotSubtotalTypes.Sum);
pivotTable.BuiltInStyle = PivotBuiltInStyles.PivotStyleDark12;
//pivot table layout
pivotTable.Layout();
workbook.SaveAs("PivotTable.xlsx");
excelEngine.ThrowNotSavedOnDestroy = false;
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("PivotData.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim pivotSheet As IWorksheet = workbook.Worksheets(1)
Dim cache As IPivotCache = workbook.PivotCaches.Add(worksheet("A1:H50"))
Dim pivotTable As IPivotTable = pivotSheet.PivotTables.Add("PivotTable1",
pivotSheet("A1"), cache)
pivotTable.Fields(4).Axis = PivotAxisTypes.Page
pivotTable.Fields(2).Axis = PivotAxisTypes.Row
pivotTable.Fields(6).Axis = PivotAxisTypes.Row
pivotTable.Fields(3).Axis = PivotAxisTypes.Column
Dim dataField As IPivotField = pivotSheet.PivotTables(0).Fields(5)
pivotTable.DataFields.Add(dataField, "Sum of Units", PivotSubtotalTypes.Sum)
pivotTable.BuiltInStyle = PivotBuiltInStyles.PivotStyleDark12
'pivot table layout
pivotTable.Layout()
workbook.SaveAs("PivotTable.xlsx")
excelEngine.ThrowNotSavedOnDestroy = False
End Using

```

UWP

```

//XlsIO supports pivot table layout in Windows Forms, WPF, ASP.NET, and
ASP.NET MVC platforms.

```

ASP.NET CORE

```

//XlsIO supports pivot table layout in Windows Forms, WPF, ASP.NET, and
ASP.NET MVC platforms.

```

XAMARIN

```

//XlsIO supports pivot table layout in Windows Forms, WPF, ASP.NET, and
ASP.NET MVC platforms.

```

Working with Pivot Charts

PivotCharts are interactive graphical representations of PivotTable data that allows rapid analysis of the displayed data. In XlsIO, **PivotCharts** are created by **ICChart** interface by setting its pivot source as **PivotTable**.

Note: XlsIO supports PivotCharts only for XLSX format.

To create a pivot table, refer [Create Pivot Table](#).

The following code snippet illustrates how to create a PivotChart.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("PivotTable.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    IPivotTable pivotTable = worksheet.PivotTables[0];
    //Adding a chart to workbook
    IChart pivotChart = workbook.Charts.Add();
    //Set PivotTable as PivotSource to the chart
    pivotChart.PivotSource = pivotTable;
    //Set PivotChart type
    pivotChart.PivotChartType = ExcelChartType.Column_Clustered;
    workbook.SaveAs("PivotChart.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = ExcelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("PivotTable.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim pivotTable As IPivotTable = worksheet.PivotTables(0)
'Adding a chart to workbook
Dim pivotChart As IChart = workbook.Charts.Add()
'Set PivotTable as PivotSource to the chart
pivotChart.PivotSource = pivotTable
'Set PivotChart type
pivotChart.PivotChartType = ExcelChartType.Column_Clustered
workbook.SaveAs("PivotChart.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
```

```

//Gets input Excel document from embedded resource collection
Stream inputStream =
assembly.GetManifestResourceStream("PivotChart.PivotTable.xlsx");
IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream);
IWorksheet worksheet = workbook.Worksheets[0];
IPivotTable pivotTable = workbook.Worksheets[1].PivotTables[0];
//Adding a chart to workbook
IChart pivotChart = workbook.Charts.Add();
//Set PivotTable as PivotSource to the chart
pivotChart.PivotSource = pivotTable;
//Set PivotChart type
pivotChart.PivotChartType = ExcelChartType.Column_Clustered;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "PivotChart";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
FileStream fileStream = new FileStream("PivotTable.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(fileStream);
IWorksheet worksheet = workbook.Worksheets[0];
IPivotTable pivotTable = worksheet.PivotTables[0];
//Adding a chart to workbook
IChart pivotChart = workbook.Charts.Add();
//Set PivotTable as PivotSource to the chart
pivotChart.PivotSource = pivotTable;
//Set PivotChart type
pivotChart.PivotChartType = ExcelChartType.Column_Clustered;
string fileName = "PivotChart.xlsx";
//Saving the workbook as stream
FileStream stream = new FileStream(fileName, FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Gets assembly

```

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from embedded resource collection
Stream inputStream =
assembly.GetManifestResourceStream("PivotChart.PivotTable.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet worksheet = workbook.Worksheets[0];
IPivotTable pivotTable = worksheet.PivotTables[0];
//Adding a chart to workbook
IChart pivotChart = workbook.Charts.Add();
//Set PivotTable as PivotSource to the chart
pivotChart.PivotSource = pivotTable;
//Set PivotChart type
pivotChart.PivotChartType = ExcelChartType.Column_Clustered;
string fileName = "PivotChart.xlsx";
//Saving the workbook as stream
MemoryStream outputStream = new MemoryStream();
workbook.SaveAs(outputStream);
//Save the stream as an Excel document and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
await DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
"application/msexcel", outputStream);
else
DependencyService.Get<ISave>().SaveAndView(fileName, "application/msexcel",
outputStream);
//Dispose the input and output stream instances
inputStream.Dispose();
outputStream.Dispose();
}

```

PivotChart Options

The following code snippet shows how to set field buttons in a pivot chart.

Note: The PivotChart properties are supported exclusively from Excel 2010 onwards.

C#

```

//Adding PivotChart to the workbook
IChart pivotChart = workbook.Charts.Add();
//Set Field Buttons
pivotChart.ShowAllFieldButtons = false;
pivotChart.ShowAxisFieldButtons = false;
pivotChart.ShowLegendFieldButtons = false;
pivotChart.ShowReportFilterFieldButtons = false;
pivotChart.ShowValueFieldButtons = false;

```

VB.NET

```

'Insert the PivotChart sheet to the workbook
Dim pivotChartSheet As IChart = workbook.Charts.Add()
'Set Field Buttons
pivotChartSheet.ShowAllFieldButtons = False
pivotChartSheet.ShowAxisFieldButtons = False

```

```
pivotChartSheet.ShowLegendFieldButtons = False  
pivotChartSheet.ShowReportFilterFieldButtons = False  
pivotChartSheet.ShowValueFieldButtons = False
```

UWP

```
//Adding PivotChart to the workbook  
IChart pivotChart = workbook.Charts.Add();  
//Set Field Buttons  
pivotChart.ShowAllFieldButtons = false;  
pivotChart.ShowAxisFieldButtons = false;  
pivotChart.ShowLegendFieldButtons = false;  
pivotChart.ShowReportFilterFieldButtons = false;  
pivotChart.ShowValueFieldButtons = false;
```

ASP.NET CORE

```
//Adding PivotChart to the workbook  
IChart pivotChart = workbook.Charts.Add();  
//Set Field Buttons  
pivotChart.ShowAllFieldButtons = false;  
pivotChart.ShowAxisFieldButtons = false;  
pivotChart.ShowLegendFieldButtons = false;  
pivotChart.ShowReportFilterFieldButtons = false;  
pivotChart.ShowValueFieldButtons = false;
```

XAMARIN

```
//Adding PivotChart to the workbook  
IChart pivotChart = workbook.Charts.Add();  
//Set Field Buttons  
pivotChart.ShowAllFieldButtons = false;  
pivotChart.ShowAxisFieldButtons = false;  
pivotChart.ShowLegendFieldButtons = false;  
pivotChart.ShowReportFilterFieldButtons = false;  
pivotChart.ShowValueFieldButtons = false;
```

Security

You can protect an anonymous user from viewing, moving, editing or deleting important data from a worksheet or workbook by [protecting a worksheet or workbook](#), with or without a password.

Protect Workbook

To keep others from making structural changes to your documents such as moving, deleting and adding sheets, you can protect the workbook in the following ways.

Encryption with password

There are two different passwords to encrypt a document.

1. **Password To Open** - This password helps to protect your workbook from unauthorized viewing or accessing.
2. **Password to Modify** - This password helps to allow give specific users permission to modify the workbook data and save changes to the file.

Read-Only Recommended – If the excel file is set as Read-only recommended, then Microsoft Excel displays a message recommending that you open the workbook as read-only when users open the excel file. This can be set with or without requiring a password to open the file.

The Following code snippets illustrate how to achieve the above options.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Encrypt the workbook with password
    workbook.PasswordToOpen = "password";
    //Set the password to modify the workbook
    workbook.SetWriteProtectionPassword("modify_password");
    //Set the workbook as read-only
    workbook.ReadOnlyRecommended = true;
    workbook.SaveAs("Encrypt.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
'Encrypt the workbook with password
workbook.PasswordToOpen = "password"
'Set the password to modify the workbook
workbook.SetWriteProtectionPassword("modify_password")
'Set the workbook as read-only
workbook.ReadOnlyRecommended = True
workbook.SaveAs("Encrypt.xlsx")
End Using
```

UWP

```
//Encrypt and Decrypt can be performed by referring .NET Standard assemblies
in UWP platform.
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Encrypt the workbook with password
    workbook.PasswordToOpen = "password";
    //Set the password to modify the workbook
    workbook.SetWriteProtectionPassword("modify_password");
    //Set the workbook as read-only
    workbook.ReadOnlyRecommended = true;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
}
```

```

savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from the FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
var file = await storageFile.OpenAsync(FileAccessMode.ReadWrite);
Stream stream = file.AsStreamForWrite();
workbook.SaveAs(stream);
await file.FlushAsync();
stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Encrypt the workbook with password
    workbook.PasswordToOpen = "password";
    //Set the password to modify the workbook
    workbook.SetWriteProtectionPassword("modify_password");
    //Set the workbook as read-only
    workbook.ReadOnlyRecommended = true;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Encrypt.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Encrypt the workbook with password
    workbook.PasswordToOpen = "password";
    //Set the password to modify the workbook
    workbook.SetWriteProtectionPassword("modify_password");
    //Set the workbook as read-only
    workbook.ReadOnlyRecommended = true;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {

```

```
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx", "application/msexcel", stream);
}
}
```

Now, the encrypted workbook can be saved. Refer [Save Excel file](#).

Opening an encrypted workbook

You can open an existing encrypted workbook (decrypting) from either the file system or the stream using the following overloads.

C#

```
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//Loads or open an existing workbook through Open method of IWorkbooks
IWorkbook workbook = excelEngine.Excel.Workbooks.Open(fileName,
ExcelParseOptions.Default, false, "password");
```

VB.NET

```
'Creates a new instance for ExcelEngine
Dim excelEngine As New ExcelEngine()
'Loads or open an existing workbook through Open method of IWorkbooks
Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Open(fileName,
ExcelParseOptions.Default, False, "password")
```

UWP

```
//Encrypt and Decrypt can be performed by referring .NET Standard assemblies in UWP platform.
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//Loads or open an existing workbook through Open method of IWorkbooks
IWorkbook workbook = excelEngine.Excel.Workbooks.Open(fileName,
ExcelParseOptions.Default, false, "password");
```

ASP.NET CORE

```
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
//Loads or open an existing workbook through Open method of IWorkbooks
IWorkbook workbook = excelEngine.Excel.Workbooks.Open(workbookStream,
ExcelParseOptions.Default, false, "password");
```

XAMARIN

```
//Creates a new instance for ExcelEngine
ExcelEngine excelEngine = new ExcelEngine();
```

```
//Loads or open an existing workbook through Open method of IWorkbooks
IWorkbook workbook = excelEngine.Excel.Workbooks.Open(fileName,
ExcelParseOptions.Default, false, "password");
```

Removing encryption

The following code illustrates how to remove a protection for an encrypted document.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = excelEngine.Excel.Workbooks.Open("Sample.xlsx",
    ExcelParseOptions.Default, true, "password");
    //Removing a protection
    workbook.PasswordToOpen = string.Empty;
    workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Open("Sample.xlsx",
ExcelParseOptions.Default, True, "password")
'Removing a protection
workbook.PasswordToOpen = String.Empty
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
//Encrypt and Decrypt can be performed by referring .NET Standard assemblies in UWP platform.
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("Security.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelParseOptions.Default, true, "password");
    IWorksheet sheet = workbook.Worksheets[0];
    //Removing a protection
    workbook.PasswordToOpen = string.Empty;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
```

```

savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from the FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
var file = await storageFile.OpenAsync(FileAccessMode.ReadWrite);
Stream stream = file.AsStreamForWrite();
workbook.SaveAs(stream);
await file.FlushAsync();
stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelParseOptions.Default, true, "password");
    //Removing a protection
    workbook.PasswordToOpen = string.Empty;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("Security.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelParseOptions.Default, true, "password");
    IWorksheet sheet = workbook.Worksheets[0];
    //Removing a protection
    workbook.PasswordToOpen = string.Empty;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)

```

```

{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output.xlsx", "application/msexcel", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx", "application/msexcel", stream);
}
}

```

Protect workbook elements

XlsIO provides options to protect and unprotect workbook elements with password. The following code example illustrates how to protect a workbook with a password.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    bool isProtectWindow = true;
    bool isProtectContent = true;
    //Protect Workbook
    workbook.Protect(isProtectWindow, isProtectContent, "password");
    workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook =
application.Workbooks.Open("ProtectWorkbook.xlsx")
Dim isProtectWindow As Boolean = True
Dim isProtectContent As Boolean = True
'protect workbook
workbook.Protect(isProtectWindow, isProtectContent, "password")
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

//Encrypt and Decrypt can be performed by referring .NET Standard assemblies
in UWP platform.
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection

```

```

Stream inputStream =
assembly.GetManifestResourceStream("Security.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelParseOptions.Default, true, "password");
IWorksheet sheet = workbook.Worksheets[0];
bool isProtectWindow = true;
bool isProtectContent = true;
//Protect Workbook
workbook.Protect(isProtectWindow, isProtectContent, "password");
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from the FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
var file = await storageFile.OpenAsync(FileAccessMode.ReadWrite);
Stream stream = file.AsStreamForWrite();
workbook.SaveAs(stream);
await file.FlushAsync();
stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
FileStream inputStream = new FileStream("ProtectWorkbook.xlsx",
FileMode.Open, FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(inputStream);
bool isProtectWindow = true;
bool isProtectContent = true;
//Protect Workbook
workbook.Protect(isProtectWindow, isProtectContent, "password");
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from an embedded resource collection
Stream inputStream =
assembly.GetManifestResourceStream("Security.Sample.xlsx");

```

```

IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelParseOptions.Default, true, "password");
IWorksheet sheet = workbook.Worksheets[0];
bool isProtectWindow = true;
bool isProtectContent = true;
//Protect Workbook
workbook.Protect(isProtectWindow, isProtectContent, "password");
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Unprotect Workbook elements

You can unprotect or remove protection for a workbook as shown below.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("ProtectedWorkbook.xlsx");
//Unprotect (unlock) Workbook using Password
workbook.Unprotect("password");
workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook =
application.Workbooks.Open("ProtectedWorkbook.xlsx")
'Unprotect (unlock) Workbook using Password
workbook.Unprotect("password")
workbook.SaveAs("Output.xlsx")
End Using

```


UWP

```

//Encrypt and Decrypt can be performed by referring .NET Standard assemblies in UWP platform.
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("Security.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelParseOptions.Default, true, "password");
    IWorksheet sheet = workbook.Worksheets[0];
    //Unprotect (unlock) Workbook using Password
    workbook.Unprotect("password");
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from the FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    var file = await storageFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream stream = file.AsStreamForWrite();
    workbook.SaveAs(stream);
    await file.FlushAsync();
    stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("ProtectedWorkbook.xlsx",
        FileMode.Open, FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    //Unprotect (unlock) Workbook using Password
    workbook.Unprotect("password");
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
        FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;

```

```

application.DefaultVersion = ExcelVersion.Excel2013;
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from an embedded resource collection
Stream inputStream =
assembly.GetManifestResourceStream("Security.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelParseOptions.Default, true, "password");
IWorksheet sheet = workbook.Worksheets[0];
//Unprotect (unlock) Workbook using Password
workbook.Unprotect("password");
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Protect Worksheet

XlsIO provides support for protecting and unprotecting elements in worksheets by using the **Protect** method of **IWorksheet**. The following code example illustrates how to protect a worksheet with a password.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Protecting the Worksheet by using a Password
sheet.Protect("syncfusion", ExcelSheetProtection.All);
workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013

```

```

Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Protecting the Worksheet by using a Password
sheet.Protect("syncfusion", ExcelSheetProtection.All)
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

//Encrypt and Decrypt can be performed by referring .NET Standard assemblies
in UWP platform.
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Protecting the Worksheet by using a Password
    sheet.Protect("syncfusion", ExcelSheetProtection.All);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from the FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    var file = await storageFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream stream = file.AsStreamForWrite();
    workbook.SaveAs(stream);
    await file.FlushAsync();
    stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Protecting the Worksheet by using a Password
    sheet.Protect("syncfusion", ExcelSheetProtection.All);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{

```

```

IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Protecting the Worksheet by using a Password
sheet.Protect("syncfusion", ExcelSheetProtection.All);
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
"application/msexcel", stream);
}
}

```

Note: By using the ExcelSheetProtection enumerator, you can set protection to the workbook elements/operations.

Chart Sheet Protection

Essential XlsIO can also provide support to protect or unprotect a chart sheet.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("sample.xlsx");
IChart chart = workbook.Charts[0];
//Protect chart sheet
chart.Protect("syncfusion", ExcelSheetProtection.All);
workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim chart As IChart = workbook.Charts(0)
'Protect chart sheet
chart.Protect("syncfusion", ExcelSheetProtection.All)

```

```
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
//Encrypt and Decrypt can be performed by referring .NET Standard assemblies in UWP platform.
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("Security.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelParseOptions.Default, true, "password");
    IChart chart = workbook.Charts[0];
    //Protect chart sheet
    chart.Protect("syncfusion", ExcelSheetProtection.All);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from the FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    var file = await storageFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream stream = file.AsStreamForWrite();
    workbook.SaveAs(stream);
    await file.FlushAsync();
    stream.Dispose();
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("ChartSheet.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IChart chart = workbook.Charts[0];
    //Protect chart sheet
    chart.Protect("syncfusion", ExcelSheetProtection.All);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
        FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("Security.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelParseOptions.Default, true, "password");
    IChart chart = workbook.Charts[0];
    //Protect chart sheet
    chart.Protect("syncfusion", ExcelSheetProtection.All);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
        "application/msexcel", stream);
    }
}

```

Un-Protect Worksheet

You can also unprotect the worksheet by using the **Unprotect** method of XlsIO. The following code example illustrates how to remove worksheet protection.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Unprotecting (unlocking) the Worksheet using the Password
    sheet.Unprotect("syncfusion");
    workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("sample.xlsx")
Dim sheet As IWorkbook = workbook.Worksheets(0)
'Unprotecting (unlocking) the Worksheet using the Password
sheet.Unprotect("syncfusion")
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

//Encrypt and Decrypt can be performed by referring .NET Standard assemblies in UWP platform.
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("Security.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelParseOptions.Default, true, "password");
    IWorksheet sheet = workbook.Worksheets[0];
    //Unprotecting (unlocking) the Worksheet using the Password
    sheet.Unprotect("syncfusion");
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from the FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    var file = await storageFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream stream = file.AsStreamForWrite();
    workbook.SaveAs(stream);
    await file.FlushAsync();
    stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Unprotecting (unlocking) the Worksheet using the Password
    worksheet.Unprotect("syncfusion");
    //Saving the workbook as stream

```

```

FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("Security.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelParseOptions.Default, true, "password");
    IWorksheet sheet = workbook.Worksheets[0];
    //Unprotecting (unlocking) the Worksheet using the Password
    sheet.Unprotect("syncfusion");
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
        "application/msexcel", stream);
    }
}

```

Removing protection of a Chart Sheet

You can remove the protection of a chart sheet as shown below.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IChart chart = workbook.Charts[0];
    //Unprotect chart sheet
}

```



```
chart.Unprotect("syncfusion");
workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim chart As IChart = workbook.Charts(0)
'Unprotect chart sheet
chart.Unprotect("syncfusion")
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
//Encrypt and Decrypt can be performed by referring .NET Standard assemblies in UWP platform.
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("Security.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelParseOptions.Default, true, "password");
    IChart chart = workbook.Charts[0];
    //Unprotect chart sheet
    chart.Unprotect("syncfusion");
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from the FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    var file = await storageFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream stream = file.AsStreamForWrite();
    workbook.SaveAs(stream);
    await file.FlushAsync();
    stream.Dispose();
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
```

```

FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(inputStream);
IChart chart = workbook.Charts[0];
//Unprotect chart sheet
chart.Unprotect("syncfusion");
//Saving the workbook as stream
FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("Security.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelParseOptions.Default, true, "password");
    IChart chart = workbook.Charts[0];
    //Unprotect chart sheet
    chart.Unprotect("syncfusion");
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android and iOS platforms. Please refer xlsio/xamarin section for respective
    //code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
        "application/msexcel", stream);
    }
}

```

Protect Cell

XlsIO supports locking and unlocking cells by using the cell's **Locked** property of **CellStyle**. This can be manipulated to make certain cells editable in a protected worksheet.

Note: By default, cells are locked. Lock or Unlock cell in an unprotected worksheet has no effect.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Unlocking a cell to edit in worksheet protection mode
    worksheet.Range["A1"].CellStyle.Locked = false;
    workbook.SaveAs("Output.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Unlocking a cell to edit in worksheet protection mode
worksheet.Range("A1").CellStyle.Locked = False
workbook.SaveAs("Output.xlsx")
End Using

```

UWP

```

//Encrypt and Decrypt can be performed by referring .NET Standard assemblies in UWP platform.
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("Security.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
        ExcelParseOptions.Default, true, "password");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Unlocking a cell to edit in worksheet protection mode
    worksheet.Range["A1"].CellStyle.Locked = false;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx" });
    //Creates a storage file from the FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    var file = await storageFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream stream = file.AsStreamForWrite();
    workbook.SaveAs(stream);
    await file.FlushAsync();
    stream.Dispose();
}

```

```
}

```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Unlocking a cell to edit in worksheet protection mode
    worksheet.Range["A1"].CellStyle.Locked = false;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("Security.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelParseOptions.Default, true, "password");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Unlocking a cell to edit in worksheet protection mode
    worksheet.Range["A1"].CellStyle.Locked = false;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
        "application/msexcel", stream);
    }
}
```

```
}
```

Note: Security features are now supported in .NET Standard 1.4 onwards.

Working with Drawing Objects

Form Controls

You can add and manipulate the text Box, option button, check box, and combo box controls into the worksheet. Enable these controls to create forms which are very user friendly.

Note: Support for Active X Form controls is not yet available.

This section explains the usage of the following [Form Controls](#).

- Text Box
- Check Box
- Combo Box
- Option Button

Text Box

The **ITextBoxShape** interface represents a text box in a worksheet. Various properties like Horizontal and Vertical Alignment, Alternative Text, Text Rotation, and so on are also supported.

The following code example illustrates how to add and manipulate a text box control.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creates a new Text Box
    ITextBoxShape textbox = sheet.TextBoxes.AddTextBox(2, 2, 30, 200);
    textbox.Text = "Text Box 1";
    textbox = sheet.TextBoxes.AddTextBox(6, 2, 30, 200);
    textbox.Text = "Text Box 2";
    //Reads a Text Box
    ITextBoxShape shapel = sheet.TextBoxes[0];
    shapel.Text = "TextBox";
    //Format the control
    shapel.Fill.ForeColor = Color.Gold;
    shapel.Fill.BackColor = Color.Black;
    shapel.Fill.Pattern = ExcelGradientPattern.Pat_90_Percent;
    //Remove a Text Box
    ITextBoxShape shape2 = sheet.TextBoxes[1];
    shape2.Remove();
    workbook.SaveAs("Textbox.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
```

```

application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creates a new Text Box
Dim textbox As ITextBoxShape = sheet.TextBoxes.AddTextBox(2, 2, 30, 200)
textbox.Text = "Text Box 1"
textbox = sheet.TextBoxes.AddTextBox(6, 2, 30, 200)
textbox.Text = "Text Box 2"
'Reads a Text Box
Dim shapel As ITextBoxShape = sheet.TextBoxes(0)
shapel.Text = "TextBox"
'Format the control
shapel.Fill.ForeColor = Color.Gold
shapel.Fill.BackColor = Color.Black
shapel.Fill.Pattern = ExcelGradientPattern.Pat_90_Percent
'Remove a Text Box
Dim shape2 As ITextBoxShape = sheet.TextBoxes(1)
shape2.Remove()
workbook.SaveAs("Textbox.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creates a new Text Box
    ITextBoxShape textbox = sheet.TextBoxes.AddTextBox(2, 2, 30, 200);
    textbox.Text = "Text Box 1";
    textbox = sheet.TextBoxes.AddTextBox(6, 2, 30, 200);
    textbox.Text = "Text Box 2";
    //Reads a Text Box
    ITextBoxShape shapel = sheet.TextBoxes[0];
    shapel.Text = "TextBox";
    //Format the control
    shapel.Fill.ForeColor = Color.FromArgb(255, 255, 215, 0);
    shapel.Fill.BackColor = Color.FromArgb(255, 0, 0, 0);
    shapel.Fill.Pattern = ExcelGradientPattern.Pat_90_Percent;
    //Remove a Text Box
    ITextBoxShape shape2 = sheet.TextBoxes[1];
    shape2.Remove();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "TextBox";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creates a new Text Box
    ITextBoxShape textbox = sheet.TextBoxes.AddTextBox(2, 2, 30, 200);
    textbox.Text = "Text Box 1";
    textbox = sheet.TextBoxes.AddTextBox(6, 2, 30, 200);
    textbox.Text = "Text Box 2";
    //Reads a Text Box
    ITextBoxShape shapel = sheet.TextBoxes[0];
    shapel.Text = "TextBox";
    //Format the control
    shapel.Fill.ForeColor = Color.Gold;
    shapel.Fill.BackColor = Color.Black;
    shapel.Fill.Pattern = ExcelGradientPattern.Pat_90_Percent;
    //Remove a Text Box
    ITextBoxShape shape2 = sheet.TextBoxes[1];
    shape2.Remove();
    //Saving the workbook as stream
    FileStream stream = new FileStream("TextBox.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creates a new Text Box
    ITextBoxShape textbox = sheet.TextBoxes.AddTextBox(2, 2, 30, 200);
    textbox.Text = "Text Box 1";
    textbox = sheet.TextBoxes.AddTextBox(6, 2, 30, 200);
    textbox.Text = "Text Box 2";
    //Reads a Text Box
    ITextBoxShape shapel = sheet.TextBoxes[0];
    shapel.Text = "TextBox";
    //Format the control
    shapel.Fill.ForeColor = Syncfusion.Drawing.Color.Gold;
    shapel.Fill.BackColor = Syncfusion.Drawing.Color.Black;
    shapel.Fill.Pattern = ExcelGradientPattern.Pat_90_Percent;
    //Remove a Text Box
    ITextBoxShape shape2 = sheet.TextBoxes[1];
    shape2.Remove();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
}
```

```

stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android, and iOS platforms. Refer to the xlsio/xamarin section for
//respective code samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("TextBo
x.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("TextBox.xlsx",
"application/msexcel", stream);
}
}

```

Check Box

ICheckBoxShape object represents a check box in a worksheet. The following code example illustrates how to insert and manipulate a check box control.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Create a check box with cell link
ICheckBoxShape checkBoxRed = sheet.CheckBoxes.AddCheckBox(2, 4, 20, 75);
checkBoxRed.Text = "Red";
checkBoxRed.CheckState = ExcelCheckState.Unchecked;
checkBoxRed.LinkedCell = sheet["B2"];
ICheckBoxShape checkBoxBlue = sheet.CheckBoxes.AddCheckBox(4, 4, 20, 75);
checkBoxBlue.Text = "Blue";
checkBoxBlue.CheckState = ExcelCheckState.Checked;
checkBoxBlue.LinkedCell = sheet["B4"];
//Read a check box
checkBoxRed = sheet.CheckBoxes[0];
checkBoxRed.CheckState = ExcelCheckState.Checked;
//Remove a check box
checkBoxBlue = sheet.CheckBoxes[1];
checkBoxBlue.Remove();
workbook.SaveAs("Checkbox.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)

```



```

'Create a check box with cell link
Dim checkBoxRed As ICheckBoxShape = sheet.CheckBoxes.AddCheckBox(2, 4, 20,
75)
checkBoxRed.Text = "Red"
checkBoxRed.CheckState = ExcelCheckState.Unchecked
checkBoxRed.LinkedCell = sheet("B2")
Dim checkBoxBlue As ICheckBoxShape = sheet.CheckBoxes.AddCheckBox(4, 4, 20,
75)
checkBoxBlue.Text = "Blue"
checkBoxBlue.CheckState = ExcelCheckState.Checked
checkBoxBlue.LinkedCell = sheet("B4")
'Read a check box
checkBoxRed = sheet.CheckBoxes(0)
checkBoxRed.CheckState = ExcelCheckState.Checked
'Remove a check box
checkBoxBlue = sheet.CheckBoxes(1)
checkBoxBlue.Remove()
workbook.SaveAs("Checkbox.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a check box with cell link
    ICheckBoxShape checkBoxRed = sheet.CheckBoxes.AddCheckBox(2, 4, 20, 75);
    checkBoxRed.Text = "Red";
    checkBoxRed.CheckState = ExcelCheckState.Unchecked;
    checkBoxRed.LinkedCell = sheet["B2"];
    ICheckBoxShape checkBoxBlue = sheet.CheckBoxes.AddCheckBox(4, 4, 20, 75);
    checkBoxBlue.Text = "Blue";
    checkBoxBlue.CheckState = ExcelCheckState.Checked;
    checkBoxBlue.LinkedCell = sheet["B4"];
    //Read a check box
    checkBoxRed = sheet.CheckBoxes[0];
    checkBoxRed.CheckState = ExcelCheckState.Checked;
    //Remove a check box
    checkBoxBlue = sheet.CheckBoxes[1];
    checkBoxBlue.Remove();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "CheckBox";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a check box with cell link
    ICheckBoxShape checkBoxRed = sheet.CheckBoxes.AddCheckBox(2, 4, 20, 75);
    checkBoxRed.Text = "Red";
    checkBoxRed.CheckState = ExcelCheckState.Unchecked;
    checkBoxRed.LinkedCell = sheet["B2"];
    ICheckBoxShape checkBoxBlue = sheet.CheckBoxes.AddCheckBox(4, 4, 20, 75);
    checkBoxBlue.Text = "Blue";
    checkBoxBlue.CheckState = ExcelCheckState.Checked;
    checkBoxBlue.LinkedCell = sheet["B4"];
    //Read a check box
    checkBoxRed = sheet.CheckBoxes[0];
    checkBoxRed.CheckState = ExcelCheckState.Checked;
    //Remove a check box
    checkBoxBlue = sheet.CheckBoxes[1];
    checkBoxBlue.Remove();
    //Saving the workbook as stream
    FileStream stream = new FileStream("CheckBox.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create a check box with cell link
    ICheckBoxShape checkBoxRed = sheet.CheckBoxes.AddCheckBox(2, 4, 20, 75);
    checkBoxRed.Text = "Red";
    checkBoxRed.CheckState = ExcelCheckState.Unchecked;
    checkBoxRed.LinkedCell = sheet["B2"];
    ICheckBoxShape checkBoxBlue = sheet.CheckBoxes.AddCheckBox(4, 4, 20, 75);
    checkBoxBlue.Text = "Blue";
    checkBoxBlue.CheckState = ExcelCheckState.Checked;
    checkBoxBlue.LinkedCell = sheet["B4"];
    //Read a check box
    checkBoxRed = sheet.CheckBoxes[0];
    checkBoxRed.CheckState = ExcelCheckState.Checked;
    //Remove a check box
    checkBoxBlue = sheet.CheckBoxes[1];
    checkBoxBlue.Remove();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
}

```

```

stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("CheckB
ox.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("CheckBox.xlsx",
"application/msexcel", stream);
}
}

```

Combo Box

IComboboxShape object represents a combo box in a worksheet. The following code example illustrates how to insert and manipulate a combo box control.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Filling Values
sheet["A2"].Text = "RGB colors";
sheet["A3"].Text = "Red";
sheet["A4"].Text = "Green";
sheet["A5"].Text = "Blue";
sheet["B5"].Text = "Selected Index";
//Create a Combo Box
IComboboxShape comboBox1 = sheet.ComboBoxes.AddComboBox(2, 3, 20, 100);
//Assign a value to the Combo Box
comboBox1.ListFillRange = sheet["A3:A5"];
comboBox1.LinkedCell = sheet["C5"];
comboBox1.SelectedIndex = 2;
//Create a Combo Box
IComboboxShape comboBox2 = sheet.ComboBoxes.AddComboBox(5, 3, 20, 100);
//Assign a value to the Combo Box
comboBox2.ListFillRange = sheet["A3:A5"];
comboBox2.LinkedCell = sheet["C5"];
comboBox2.SelectedIndex = 1;
//Read a Combo Box
comboBox1 = sheet.ComboBoxes[0];
comboBox1.SelectedIndex = 1;
//Remove a Combo Box
comboBox2 = sheet.ComboBoxes[1];
comboBox2.Remove();
workbook.SaveAs("Combobox.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Filling Values
sheet("A2").Text = "RGB colors"
sheet("A3").Text = "Red"
sheet("A4").Text = "Green"
sheet("A5").Text = "Blue"
sheet("B5").Text = "Selected Index"
'Create a Combo Box
Dim comboBox1 As IComboBoxShape = sheet.ComboBoxes.AddComboBox(2, 3, 20,
100)
'Assign a value to the Combo Box
comboBox1.ListFillRange = sheet("A3:A5")
comboBox1.LinkedCell = sheet("C5")
comboBox1.SelectedIndex = 2
'Create a Combo Box
Dim comboBox2 As IComboBoxShape = sheet.ComboBoxes.AddComboBox(5, 3, 20,
100)
'Assign a value to the Combo Box
comboBox2.ListFillRange = sheet("A3:A5")
comboBox2.LinkedCell = sheet("C5")
comboBox2.SelectedIndex = 1
'Read a Combo Box
comboBox1 = sheet.ComboBoxes(0)
comboBox1.SelectedIndex = 1
'Remove a Combo Box
comboBox2 = sheet.ComboBoxes(1)
comboBox2.Remove()
workbook.SaveAs("Combobox.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Filling Values
sheet["A2"].Text = "RGB colors";
sheet["A3"].Text = "Red";
sheet["A4"].Text = "Green";
sheet["A5"].Text = "Blue";
sheet["B5"].Text = "Selected Index";
//Create a Combo Box
IComboBoxShape comboBox1 = sheet.ComboBoxes.AddComboBox(2, 3, 20, 100);
//Assign a value to the Combo Box
comboBox1.ListFillRange = sheet["A3:A5"];
comboBox1.LinkedCell = sheet["C5"];

```

```

comboBox1.SelectedIndex = 2;
//Create a Combo Box
IComboBoxShape comboBox2 = sheet.ComboBoxes.AddComboBox(5, 3, 20, 100);
//Assign a value to the Combo Box
comboBox2.ListFillRange = sheet["A3:A5"];
comboBox2.LinkedCell = sheet["C5"];
comboBox2.SelectedIndex = 1;
//Read a Combo Box
comboBox1 = sheet.ComboBoxes[0];
comboBox1.SelectedIndex = 1;
//Remove a Combo Box
comboBox2 = sheet.ComboBoxes[1];
comboBox2.Remove();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "ComboBox";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Filling Values
    sheet["A2"].Text = "RGB colors";
    sheet["A3"].Text = "Red";
    sheet["A4"].Text = "Green";
    sheet["A5"].Text = "Blue";
    sheet["B5"].Text = "Selected Index";
    //Create a Combo Box
    IComboBoxShape comboBox1 = sheet.ComboBoxes.AddComboBox(2, 3, 20, 100);
    //Assign a value to the Combo Box
    comboBox1.ListFillRange = sheet["A3:A5"];
    comboBox1.LinkedCell = sheet["C5"];
    comboBox1.SelectedIndex = 2;
    //Create a Combo Box
    IComboBoxShape comboBox2 = sheet.ComboBoxes.AddComboBox(5, 3, 20, 100);
    //Assign a value to the Combo Box
    comboBox2.ListFillRange = sheet["A3:A5"];
    comboBox2.LinkedCell = sheet["C5"];
    comboBox2.SelectedIndex = 1;
    //Read a Combo Box
    comboBox1 = sheet.ComboBoxes[0];
    comboBox1.SelectedIndex = 1;
    //Remove a Combo Box
    comboBox2 = sheet.ComboBoxes[1];
}

```

```

comboBox2.Remove();
//Saving the workbook as stream
FileStream stream = new FileStream("ComboBox.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Filling Values
    sheet["A2"].Text = "RGB colors";
    sheet["A3"].Text = "Red";
    sheet["A4"].Text = "Green";
    sheet["A5"].Text = "Blue";
    sheet["B5"].Text = "Selected Index";
    //Create a Combo Box
    IComboBoxShape comboBox1 = sheet.ComboBoxes.AddComboBox(2, 3, 20, 100);
    //Assign a value to the Combo Box
    comboBox1.ListFillRange = sheet["A3:A5"];
    comboBox1.LinkedCell = sheet["C5"];
    comboBox1.SelectedIndex = 2;
    //Create a Combo Box
    IComboBoxShape comboBox2 = sheet.ComboBoxes.AddComboBox(5, 3, 20, 100);
    //Assign a value to the Combo Box
    comboBox2.ListFillRange = sheet["A3:A5"];
    comboBox2.LinkedCell = sheet["C5"];
    comboBox2.SelectedIndex = 1;
    //Read a Combo Box
    comboBox1 = sheet.ComboBoxes[0];
    comboBox1.SelectedIndex = 1;
    //Remove a Combo Box
    comboBox2 = sheet.ComboBoxes[1];
    comboBox2.Remove();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("ComboB
ox.xlsx", "application/msexcel", stream);
    }
    else
    {

```

```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("ComboBox.xlsx",
"application/msexcel", stream);
}
}
```

Option Button

IOptionButtonShape object represents an option button in a worksheet. The following code example illustrates how to insert and manipulate an option button control.

Note: XlsIO provides Option button support only for XLSX format.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create an Option Button
    IOptionButtonShape optionButton1 = sheet.OptionButtons.AddOptionButton(2,
3);
    //Assign a value to the Option Button
    optionButton1.Text = "Fed Ex";
    //Format the control
    optionButton1.Fill.FillType = ExcelFillType.SolidColor;
    optionButton1.Fill.ForeColor = Color.Yellow;
    //Change the check state
    optionButton1.CheckState = ExcelCheckState.Checked;
    //Create an Option Button
    IOptionButtonShape optionButton2 = sheet.OptionButtons.AddOptionButton(5,
3);
    //Assign a value to the Option Button
    optionButton2.Text = "DHL";
    //Read an Option Button
    optionButton1 = sheet.OptionButtons[0];
    optionButton1.CheckState = ExcelCheckState.Unchecked;
    //Remove an Option Button
    optionButton2 = sheet.OptionButtons[1];
    optionButton2.Remove();
    workbook.SaveAs("OptionButton.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Create an Option Button
Dim optionButton1 As IOptionButtonShape =
sheet.OptionButtons.AddOptionButton(2, 3)
'Assign a value to the Option Button
optionButton1.Text = "Fed Ex"
'Format the control
```

```

optionButton1.Fill.FillType = ExcelFillType.SolidColor
optionButton1.Fill.ForeColor = Color.Yellow
'Change the check state
optionButton1.CheckState = ExcelCheckState.Checked
'Create an Option Button
Dim optionButton2 As IOptionButtonShape =
sheet.OptionButtons.AddOptionButton(5, 3)
'Assign a value to the Option Button
optionButton2.Text = "DHL"
'Read an Option Button
optionButton1 = sheet.OptionButtons(0)
optionButton1.CheckState = ExcelCheckState.Unchecked
'Remove an Option Button
optionButton2 = sheet.OptionButtons(1)
optionButton2.Remove()
workbook.SaveAs("OptionButton.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create an Option Button
    IOptionButtonShape optionButton1 = sheet.OptionButtons.AddOptionButton(2,
    3);
    //Assign a value to the Option Button
    optionButton1.Text = "Fed Ex";
    //Format the control
    optionButton1.Fill.FillType = ExcelFillType.SolidColor;
    optionButton1.Fill.ForeColor = Color.FromArgb(255, 255, 255, 0);
    //Change the check state
    optionButton1.CheckState = ExcelCheckState.Checked;
    //Create an Option Button
    IOptionButtonShape optionButton2 = sheet.OptionButtons.AddOptionButton(5,
    3);
    //Assign a value to the Option Button
    optionButton2.Text = "DHL";
    //Read an Option Button
    optionButton1 = sheet.OptionButtons[0];
    optionButton1.CheckState = ExcelCheckState.Unchecked;
    //Remove an Option Button
    optionButton2 = sheet.OptionButtons[1];
    optionButton2.Remove();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "OptionButton";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
}

```



```
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create an Option Button
    IOptionButtonShape optionButton1 = sheet.OptionButtons.AddOptionButton(2,
    3);
    //Assign a value to the Option Button
    optionButton1.Text = "Fed Ex";
    //Format the control
    optionButton1.Fill.FillType = ExcelFillType.SolidColor;
    optionButton1.Fill.ForeColor = Color.Yellow;
    //Change the check state
    optionButton1.CheckState = ExcelCheckState.Checked;
    //Create an Option Button
    IOptionButtonShape optionButton2 = sheet.OptionButtons.AddOptionButton(5,
    3);
    //Assign a value to the Option Button
    optionButton2.Text = "DHL";
    //Read an Option Button
    optionButton1 = sheet.OptionButtons[0];
    optionButton1.CheckState = ExcelCheckState.Unchecked;
    //Remove an Option Button
    optionButton2 = sheet.OptionButtons[1];
    optionButton2.Remove();
    //Saving the workbook as stream
    FileStream stream = new FileStream("OptionButton.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Create an Option Button
    IOptionButtonShape optionButton1 = sheet.OptionButtons.AddOptionButton(2,
    3);
    //Assign a value to the Option Button
    optionButton1.Text = "Fed Ex";
    //Format the control
    optionButton1.Fill.FillType = ExcelFillType.SolidColor;
    optionButton1.Fill.ForeColor = Syncfusion.Drawing.Color.Yellow;
    //Change the check state
```

```

optionButton1.CheckState = ExcelCheckState.Checked;
//Create an Option Button
IOptionButtonShape optionButton2 = sheet.OptionButtons.AddOptionButton(5,
3);
//Assign a value to the Option Button
optionButton2.Text = "DHL";
//Read an Option Button
optionButton1 = sheet.OptionButtons[0];
optionButton1.CheckState = ExcelCheckState.Unchecked;
//Remove an Option Button
optionButton2 = sheet.OptionButtons[1];
optionButton2.Remove();
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Option
Button.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("OptionButton.xlsx"
, "application/msexcel", stream);
}
}

```

Comments

ICommentShape object represents a [comment](#) in a worksheet. You can insert both **Regular** and **Rich Text** comments. The following code example illustrates how to insert and manipulate comments.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Adding comments to a cell
sheet.Range["A1"].AddComment().Text = "Comments";
//Add Rich Text Comments
IRange range = sheet.Range["A6"];
range.AddComment().RichText.Text = "RichText";
IRichTextString richText = range.Comment.RichText;
//Formatting first 4 characters
IFont redFont = workbook.CreateFont();
redFont.Bold = true;
redFont.Color = ExcelKnownColors.Red;
richText.SetFont(0, 3, redFont);
}

```

```
workbook.SaveAs("Comments.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Adding comments to a cell
sheet.Range("A1").AddComment().Text = "Comments"
'Add Rich Text Comments
Dim range As IRange = sheet.Range("A6")
range.AddComment().RichText.Text = "RichText"
Dim richText As IRichTextString = range.Comment.RichText
'Formatting first 4 characters
Dim redFont As IFont = workbook.CreateFont()
redFont.Bold = True
redFont.Color = ExcelKnownColors.Red
richText.SetFont(0, 3, redFont)
workbook.SaveAs("Comments.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Adding comments to a cell
sheet.Range["A1"].AddComment().Text = "Comments";
//Add Rich Text Comments
IRange range = sheet.Range["A6"];
range.AddComment().RichText.Text = "RichText";
IRichTextString richText = range.Comment.RichText;
//Formatting first 4 characters
IFont redFont = workbook.CreateFont();
redFont.Bold = true;
redFont.Color = ExcelKnownColors.Red;
richText.SetFont(0, 3, redFont);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Comments";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Adding comments to a cell
    sheet.Range["A1"].AddComment().Text = "Comments";
    //Add Rich Text Comments
    IRange range = sheet.Range["A6"];
    range.AddComment().RichText.Text = "RichText";
    IRichTextString richText = range.Comment.RichText;
    //Formatting first 4 characters
    IFont redFont = workbook.CreateFont();
    redFont.Bold = true;
    redFont.Color = ExcelKnownColors.Red;
    richText.SetFont(0, 3, redFont);
    //Saving the workbook as stream
    FileStream stream = new FileStream("Comments.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Adding comments to a cell
    sheet.Range["A1"].AddComment().Text = "Comments";
    //Add Rich Text Comments
    IRange range = sheet.Range["A6"];
    range.AddComment().RichText.Text = "RichText";
    IRichTextString richText = range.Comment.RichText;
    //Formatting first 4 characters
    IFont redFont = workbook.CreateFont();
    redFont.Bold = true;
    redFont.Color = ExcelKnownColors.Red;
    richText.SetFont(0, 3, redFont);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android, and iOS platforms. Refer to the xlsio/xamarin section for
    respective code samples
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {

```

```

Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Comments.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Comments.xlsx", "application/msexcel", stream);
}
}

```

You can also fill the comments with various types of fills by using the **IFill** interface. Following code example illustrates how to fill the comment shape with a TwoColor gradient.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Adding comments to a cell
    sheet.Range["A1"].AddComment().Text = "Comments";
    //Accessing existing comment
    ICommentShape shape = sheet.Range["A1"].Comment;
    //Format the comment
    shape.Fill.TwoColorGradient();
    shape.Fill.GradientStyle = ExcelGradientStyle.Horizontal;
    shape.Fill.GradientColorType = ExcelGradientColor.TwoColor;
    shape.Fill.ForeColorIndex = ExcelKnownColors.Red;
    shape.Fill.BackColorIndex = ExcelKnownColors.White;
    workbook.SaveAs("FormatComments.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Adding comments to a cell
sheet.Range("A1").AddComment().Text = "Comments"
'Accessing existing comment
Dim shape As ICommentShape = sheet.Range("A1").Comment
'Format the comment
shape.Fill.TwoColorGradient()
shape.Fill.GradientStyle = ExcelGradientStyle.Horizontal
shape.Fill.GradientColorType = ExcelGradientColor.TwoColor
shape.Fill.ForeColorIndex = ExcelKnownColors.Red
shape.Fill.BackColorIndex = ExcelKnownColors.White
workbook.SaveAs("FormatComments.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Adding comments to a cell
    sheet.Range["A1"].AddComment().Text = "Comments";
    //Accessing existing comment
    ICommentShape shape = sheet.Range["A1"].Comment;
    //Format the comment
    shape.Fill.TwoColorGradient();
    shape.Fill.GradientStyle = ExcelGradientStyle.Horizontal;
    shape.Fill.GradientColorType = ExcelGradientColor.TwoColor;
    shape.Fill.ForeColorIndex = ExcelKnownColors.Red;
    shape.Fill.BackColorIndex = ExcelKnownColors.White;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "FormatComments";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Adding comments to a cell
    sheet.Range["A1"].AddComment().Text = "Comments";
    //Accessing existing comment
    ICommentShape shape = sheet.Range["A1"].Comment;
    //Format the comment
    shape.Fill.TwoColorGradient();
    shape.Fill.GradientStyle = ExcelGradientStyle.Horizontal;
    shape.Fill.GradientColorType = ExcelGradientColor.TwoColor;
    shape.Fill.ForeColorIndex = ExcelKnownColors.Red;
    shape.Fill.BackColorIndex = ExcelKnownColors.White;
    //Saving the workbook as stream
    FileStream stream = new FileStream("FormatComments.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())

```

```

{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Adding comments to a cell
    sheet.Range["A1"].AddComment().Text = "Comments";
    //Accessing existing comment
    ICommentShape shape = sheet.Range["A1"].Comment;
    //Format the comment
    shape.Fill.TwoColorGradient();
    shape.Fill.GradientStyle = ExcelGradientStyle.Horizontal;
    shape.Fill.GradientColorType = ExcelGradientColor.TwoColor;
    shape.Fill.ForeColorIndex = ExcelKnownColors.Red;
    shape.Fill.BackColorIndex = ExcelKnownColors.White;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Format
        Comments.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("FormatComments.xls
        x", "application/msexcel", stream);
    }
}

```

Show or Hide Excel Comments

Comments in an Excel document can be shown or hidden using **IsVisible** property. The following code example illustrates this.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding comments in the worksheet
    worksheet.Range["E5"].AddComment();
    worksheet.Range["E15"].AddComment();
    //Adding text in comments
    worksheet.Comments[0].Text = "Comment1";
    worksheet.Comments[1].Text = "Comment2";
    //Show comment
    worksheet.Comments[0].IsVisible = true;
}

```

```
//Hide comment
worksheet.Comments[1].IsVisible = false;
workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding comments in the worksheet
worksheet.Range("E5").AddComment()
worksheet.Range("E15").AddComment()
'Adding text in comments
worksheet.Comments(0).Text = "Comment1"
worksheet.Comments(1).Text = "Comment2"
'Show comment
worksheet.Comments(0).IsVisible = True
'Hide comment
worksheet.Comments(1).IsVisible = False
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Adding comments in the worksheet
worksheet.Range["E5"].AddComment();
worksheet.Range["E15"].AddComment();
//Adding text in comments
worksheet.Comments[0].Text = "Comment1";
worksheet.Comments[1].Text = "Comment2";
//Show comment
worksheet.Comments[0].IsVisible = true;
//Hide comment
worksheet.Comments[1].IsVisible = false;
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```


ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding comments in the worksheet
    worksheet.Range["E5"].AddComment();
    worksheet.Range["E15"].AddComment();
    //Adding text in comments
    worksheet.Comments[0].Text = "Comment1";
    worksheet.Comments[1].Text = "Comment2";
    //Show comment
    worksheet.Comments[0].IsVisible = true;
    //Hide comment
    worksheet.Comments[1].IsVisible = false;
    //Saving the workbook as stream
    FileStream stream = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding comments in the worksheet
    worksheet.Range["E5"].AddComment();
    worksheet.Range["E15"].AddComment();
    //Adding text in comments
    worksheet.Comments[0].Text = "Comment1";
    worksheet.Comments[1].Text = "Comment2";
    //Show comment
    worksheet.Comments[0].IsVisible = true;
    //Hide comment
    worksheet.Comments[1].IsVisible = false;
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output
        .xlsx", "application/msexcel", stream);
    }
}

```

```

}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx",
    "application/msexcel", stream);
}
}

```

Following code snippets illustrates how to remove all the comments in existing worksheet.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Comments.xlsx");
    IWorksheet sheet = workbook.Worksheets[0];
    //Remove all the comments in worksheet
    sheet.Comments.Clear();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs("RemoveComments.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Comments.xlsx")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Remove all the comments in worksheet
sheet.Comments.Clear()
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("RemoveComments.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Remove all the comments in worksheet
    worksheet.Comments.Clear();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "RemoveComments";
}

```

```

savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
workbook.Version = ExcelVersion.Excel2013;
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream inputStream = new FileStream("Comments.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Remove all the comments in worksheet
    worksheet.Comments.Clear();
    //Saving the workbook as stream
    FileStream stream = new FileStream("RemoveComments.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    /"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Com
    ments.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Remove all the comments in worksheet
    worksheet.Comments.Clear();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.Version = ExcelVersion.Excel2013;
    workbook.SaveAs(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Remove
        Comments.xlsx", "application/msexcel", stream);
    }
}

```

```

}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("RemoveComments.xls
x", "application/msexcel", stream);
}
}

```

AutoShapes

The **IShape** interface represents an [AutoShape](#) in an Excel workbook.

To learn more about various AutoShape types supported in XlsIO, refer to the **AutoShapeType** enumeration in API section.

The following code example illustrates how to insert and format AutoShapes.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding an AutoShape
    IShape shape1 =
        worksheet.Shapes.AddAutoShapes(AutoShapeType.RoundedRectangle, 2, 7, 60,
        192);
    IShape shape2 = worksheet.Shapes.AddAutoShapes(AutoShapeType.CircularArrow,
    8, 7, 60, 192);
    //Set the value inside the shape
    shape1.TextFrame.TextRange.Text = "AutoShape";
    //Format the shape
    shape1.Fill.ForeColorIndex = ExcelKnownColors.Light_blue;
    shape1.TextFrame.VerticalAlignment = ExcelVerticalAlignment.MiddleCentered;
    //Read an AutoShape
    shape1 = worksheet.Shapes[0];
    shape1.TextFrame.TextRange.Text = "RoundedRectangle";
    //Remove an AutoShape
    shape2 = worksheet.Shapes[1];
    shape2.Remove();
    workbook.SaveAs("Autoshapes.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding an AutoShape
Dim shape1 As IShape =
    worksheet.Shapes.AddAutoShapes(AutoShapeType.RoundedRectangle, 2, 7, 60,
    192)

```

```

Dim shape2 As IShape =
worksheet.Shapes.AddAutoShapes(AutoShapeType.CircularArrow, 8, 7, 60, 192)
'Set the value inside the shape.
shape1.TextFrame.TextRange.Text = "AutoShape"
'Format the shape
shape1.Fill.ForeColorIndex = ExcelKnownColors.Light_blue
shape1.TextFrame.VerticalAlignment = ExcelVerticalAlignment.MiddleCentered
'Read an AutoShape
shape1 = worksheet.Shapes(0)
shape1.TextFrame.TextRange.Text = "RoundedRectangle"
'Remove an AutoShape
shape2 = worksheet.Shapes(1)
shape2.Remove()
workbook.SaveAs("Autoshapes.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding an AutoShape
    IShape shape1 =
    worksheet.Shapes.AddAutoShapes(AutoShapeType.RoundedRectangle, 2, 7, 60,
    192);
    IShape shape2 = worksheet.Shapes.AddAutoShapes(AutoShapeType.CircularArrow,
    8, 7, 60, 192);
    //Set the value inside the shape
    shape1.TextFrame.TextRange.Text = "AutoShape";
    //Format the shape
    shape1.Fill.ForeColorIndex = ExcelKnownColors.Light_blue;
    shape1.TextFrame.VerticalAlignment = ExcelVerticalAlignment.MiddleCentered;
    //Read an AutoShape
    shape1 = worksheet.Shapes[0];
    shape1.TextFrame.TextRange.Text = "RoundedRectangle";
    //Remove an AutoShape
    shape2 = worksheet.Shapes[1];
    shape2.Remove();
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "AutoShapes";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())

```

```

{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding an AutoShape
    IShape shape1 =
        worksheet.Shapes.AddAutoShapes(AutoShapeType.RoundedRectangle, 2, 7, 60,
        192);
    IShape shape2 = worksheet.Shapes.AddAutoShapes(AutoShapeType.CircularArrow,
    8, 7, 60, 192);
    //Set the value inside the shape
    shape1.TextFrame.TextRange.Text = "AutoShape";
    //Format the shape
    shape1.Fill.ForeColorIndex = ExcelKnownColors.Light_blue;
    shape1.TextFrame.VerticalAlignment = ExcelVerticalAlignment.MiddleCentered;
    //Read an AutoShape
    shape1 = worksheet.Shapes[0];
    shape1.TextFrame.TextRange.Text = "RoundedRectangle";
    //Remove an AutoShape
    shape2 = worksheet.Shapes[1];
    shape2.Remove();
    //Saving the workbook as stream
    FileStream stream = new FileStream("AutoShapes.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding an AutoShape
    IShape shape1 =
        worksheet.Shapes.AddAutoShapes(AutoShapeType.RoundedRectangle, 2, 7, 60,
        192);
    IShape shape2 = worksheet.Shapes.AddAutoShapes(AutoShapeType.CircularArrow,
    8, 7, 60, 192);
    //Set the value inside the shape
    shape1.TextFrame.TextRange.Text = "AutoShape";
    //Format the shape
    shape1.Fill.ForeColorIndex = ExcelKnownColors.Light_blue;
    shape1.TextFrame.VerticalAlignment = ExcelVerticalAlignment.MiddleCentered;
    //Read an AutoShape
    shape1 = worksheet.Shapes[0];
    shape1.TextFrame.TextRange.Text = "RoundedRectangle";
    //Remove an AutoShape
    shape2 = worksheet.Shapes[1];
    shape2.Remove();
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
}

```

```

workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android, and iOS platforms. Refer to the xlsio/xamarin section for
//respective code samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("AutoSh
apes.xlsx", "application/msexcel", stream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("AutoShapes.xlsx",
"application/msexcel", stream);
}
}

```

Group Shapes

Group shape can be used to work with multiple shapes at a single instant.

For example, you can change positions, size or set colors for all shapes at the same time through a single group shape.

Create Group Shapes

The shapes in the worksheet can be grouped into a single shape by creating group shape in XlsIO using IGroupShape interface.

The following code example illustrates how to create a group shape.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("GroupShape.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
IShapes shapes = worksheet.Shapes;
// Select the shapes you want to group
IShape[] groupItems = new IShape[] { shapes[0], shapes[1] };
// Group the selected shapes
IGroupShape GroupShape = shapes.Group(groupItems);
workbook.SaveAs("GroupShape.xlsx");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("GroupShape.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim shapes As IShapes = worksheet.Shapes
' Select the shapes you want to group
Dim groupItems As IShape() = New IShape() {shapes(0), shapes(1)}
' Group the selected shapes

```

```
Dim GroupShape As IGroupShape = shapes.Group(groupItems)
workbook.SaveAs("GroupShape.xlsx")
workbook.Close()
excelEngine.Dispose()
```

UWP

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile openFile = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(openFile);
IWorksheet worksheet = workbook.Worksheets[0];
IShapes shapes = worksheet.Shapes;
// Select the shapes you want to group
IShape[] groupItems = new IShape[] { shapes[0], shapes[1] };
// Group the selected shapes
IGroupShape GroupShape = shapes.Group(groupItems);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "GroupShape";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
workbook.Close();
excelEngine.Dispose();
```

ASP.NET CORE

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
FileStream inputStream = new FileStream("GroupShape.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet worksheet = workbook.Worksheets[0];
IShapes shapes = worksheet.Shapes;
// Select the shapes you want to group
IShape[] groupItems = new IShape[] { shapes[0], shapes[1] };
// Group the selected shapes
IGroupShape GroupShape = shapes.Group(groupItems);
FileStream file = new FileStream("GroupShape.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(file);
file.Dispose();
workbook.Close();
excelEngine.Dispose();
```


XAMARIN

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream = assembly.GetManifestResourceStream("GroupShape.xlsx");
IWorkbook workbook = application.Workbooks.Open(fileStream);
IWorksheet worksheet = workbook.Worksheets[0];
IShapes shapes = worksheet.Shapes;
// Select the shapes you want to group
IShape[] groupItems = new IShape[] { shapes[0], shapes[1] };
// Group the selected shapes
IGroupShape GroupShape = shapes.Group(groupItems);
workbook.SaveAs("GroupShape.xlsx");
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
workbook.Close();
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("GroupS
hape.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("GroupShape.xlsx",
"application/msexcel", stream);
}
excelEngine.Dispose();

```

Ungroup shapes

Group shape can be ungrouped, and its inner shapes are added to worksheet as an individual shape.

The following code example illustrates how to ungroup the shape.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("GroupShape.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
IShapes shapes = worksheet.Shapes;
// Ungroup the selected specified group shape
shapes.Ungroup(shapes[0] as IGroupShape);
workbook.SaveAs("GroupShape.xlsx");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("GroupShape.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim shapes As IShapes = worksheet.Shapes
' Ungroup the selected specified group shape
shapes.Ungroup(TryCast(shapes(0), IGroupShape))
workbook.SaveAs("GroupShape.xlsx")
workbook.Close()
excelEngine.Dispose()

```

UWP

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile openFile = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(openFile);
IWorksheet worksheet = workbook.Worksheets[0];
IShapes shapes = worksheet.Shapes;
// Ungroup the selected specified group shape
shapes.Ungroup(shapes[0] as IGroupShape);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "GroupShape";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
workbook.Close();
excelEngine.Dispose();

```

ASP.NET CORE

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
FileStream inputStream = new FileStream("GroupShape.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet worksheet = workbook.Worksheets[0];
IShapes shapes = worksheet.Shapes;
// Ungroup the selected specified group shape
shapes.Ungroup(shapes[0] as IGroupShape);
FileStream file = new FileStream("GroupShape.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(file);
file.Dispose();
workbook.Close();

```

```
excelEngine.Dispose();
```

XAMARIN

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream = assembly.GetManifestResourceStream("GroupShape.xlsx");
IWorkbook workbook = application.Workbooks.Open(fileStream);
IWorksheet worksheet = workbook.Worksheets[0];
IShapes shapes = worksheet.Shapes;
// Ungroup the selected specified group shape
shapes.Ungroup(shapes[0] as IGroupShape);
workbook.SaveAs("GroupShape.xlsx");
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
workbook.Close();
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("GroupS
hape.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("GroupShape.xlsx",
"application/msexcel", stream);
}
excelEngine.Dispose();
```

Ungroup all shapes

When ungrouping the group shape, its immediate inner shapes only be ungrouped. Ungroup the group shape and its all the inner shapes can be possible in XlsIO.

In Ungroup method, “isAll” boolean value indicates whether the group inner shape will be grouped or not.

The following code example illustrates how to ungroup the group shape and its inner shapes.

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("GroupShape.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
IShapes shapes = worksheet.Shapes;
// Ungroup the selected specified group shape and its inner shapes by
// specifying isAll property
shapes.Ungroup(shapes[0] as IGroupShape, true);
workbook.SaveAs("GroupShape.xlsx");
workbook.Close();
```

```
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("GroupShape.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim shapes As IShapes = worksheet.Shapes
' Ungroup the selected specified group shape and its inner shapes by
specifying isAll property
shapes.Ungroup(TryCast(shapes(0), IGroupShape), True)
workbook.SaveAs("GroupShape.xlsx")
workbook.Close()
excelEngine.Dispose()
```

UWP

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsx");
openPicker.FileTypeFilter.Add(".xls");
StorageFile openFile = await openPicker.PickSingleFileAsync();
//Opens the workbook
IWorkbook workbook = await application.Workbooks.OpenAsync(openFile);
IWorksheet worksheet = workbook.Worksheets[0];
IShapes shapes = worksheet.Shapes;
// Ungroup the selected specified group shape and its inner shapes by
specifying isAll property
shapes.Ungroup(shapes[0] as IGroupShape, true);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "GroupShape";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
workbook.Close();
excelEngine.Dispose();
```

ASP.NET CORE

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
FileStream inputStream = new FileStream("GroupShape.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet worksheet = workbook.Worksheets[0];
IShapes shapes = worksheet.Shapes;
```

```
// Ungroup the selected specified group shape and its inner shapes by
specifying isAll property
shapes.Ungroup(shapes[0] as IGroupShape, true);
FileStream file = new FileStream("GroupShape.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(file);
file.Dispose();
workbook.Close();
excelEngine.Dispose();
```

XAMARIN

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream fileStream = assembly.GetManifestResourceStream("GroupShape.xlsx");
IWorkbook workbook = application.Workbooks.Open(fileStream);
IWorksheet worksheet = workbook.Worksheets[0];
IShapes shapes = worksheet.Shapes;
// Ungroup the selected specified group shape and its inner shapes by
specifying isAll property
shapes.Ungroup(shapes[0] as IGroupShape, true);
workbook.SaveAs("GroupShape.xlsx");
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
workbook.Close();
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android and iOS platforms. Please refer xlsio/xamarin section for respective
code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("GroupS
hape.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("GroupShape.xlsx",
"application/msexcel", stream);
}
excelEngine.Dispose();
```

OLE Objects

IOleObject object represents an [OLE Object](#) in a worksheet.

Note: XlsIO supports OLE Objects for XLSX format in Windows, ASP.NET, and WPF platforms only.

OLE Objects and Linking Types

XlsIO supports two types of association of the objects:

- Linked objects
- Embedded objects

1. Linked Objects

Linked objects remains as a separate files. When the file is opened in another machine, then the linked object should be in the same location as created.

The following sample code illustrates how to link an OLE Object to an Excel document.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    Image image = Image.FromFile("image.png");
    //Select the object, image for the display icon and the type of the
OLEObject to insert
    IOleObject ole2 = sheet.OleObjects.Add("Document.docx", image,
    OleLinkType.Link);
    workbook.SaveAs("LinkedObjects.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim image As Image = Image.FromFile("image.png")
'Select the object, image for the display icon and the type of the OLEObject
to insert
Dim ole2 As IOleObject = sheet.OleObjects.Add("Document.docx", image,
OleLinkType.Link)
workbook.SaveAs("LinkedObjects.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream imageStream =
    assembly.GetManifestResourceStream("UWP.Data.image.png");
    //Get image from stream
    Syncfusion.XlsIO.Image image =
    Syncfusion.XlsIO.Image.FromStream(imageStream);
    //Add ole object
    IOleObject oleObject =
    worksheet.OleObjects.AddLink(@"..\..\Data\Document.docx", image);
    //Initializes FileSavePicker
}
```

```

FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "LinkedObjects";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create image stream
    FileStream imageStream = new FileStream("image.png", FileMode.Open);
    //Get image from stream
    Image image = Image.FromStream(imageStream);
    //Add ole object
    IOleObject oleObject =
    worksheet.OleObjects.AddLink(@"..\..\Data\Document.docx", image);
    //Saving the workbook as stream
    FileStream stream = new FileStream("LinkedObjects.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream imageStream =
    assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Ima
    ge.png");
    //Get image from stream
    Syncfusion.Drawing.Image image =
    Syncfusion.Drawing.Image.FromStream(imageStream);
    //Add ole object
    IOleObject oleObject =
    worksheet.OleObjects.AddLink(@"..\..\Data\Document.docx", image);
    //Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
}

```

```
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android, and iOS platforms. Refer to the xlsio/xamarin section for
//respective code samples
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Linked
Objects.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("LinkedObjects.xlsx
", "application/msexcel", stream);
}
}
```

2. Embedded Objects

Embedded objects are stored in the document. When the file is opened in another machine, the embedded object can be viewed without having access to the original data.

The following sample code illustrates how to embed an OLE Object to an Excel document.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
Image image = Image.FromFile("image.png");
// Select the object, image for the display icon and the type of the
OLEObject to insert
IOleObject ole1 = sheet.OleObjects.Add("Test.pptx", image,
OleLinkType.Embed);
workbook.SaveAs("EmbeddedObjects.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim image As Image = Image.FromFile("image.png")
'Select the object, image for the display icon and the type of the OLEObject
to insert
Dim ole1 As IOleObject = sheet.OleObjects.Add("Test.pptx", image,
OleLinkType.Embed)
workbook.SaveAs("EmbeddedObjects.xlsx")
End Using
```


UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    // "App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
    assembly.GetManifestResourceStream("UWP.Data.Test.pptx");
    Stream imageStream =
    assembly.GetManifestResourceStream("UWP.Data.image.png");
    //Get image from stream
    Syncfusion.XlsIO.Image image =
    Syncfusion.XlsIO.Image.FromStream(imageStream);
    //Add ole object
    IOleObject oleObject = worksheet.OleObjects.Add(inputStream, image,
    OleObjectType.PowerPointPresentation);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "EmbeddedObjects";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Create file stream and image stream
    FileStream inputStream = new FileStream("Test.pptx", FileMode.Open);
    FileStream imageStream = new FileStream("image.png", FileMode.Open);
    //Get image from stream
    Image image = Image.FromStream(imageStream);
    //Add ole object
    IOleObject oleObject = worksheet.OleObjects.Add(inputStream, image,
    OleObjectType.PowerPointPresentation);
    //Saving the workbook as stream
    FileStream stream = new FileStream("EmbeddedObjects.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Tes
t.pptx");
    Stream imageStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Ima
ge.png");
    ///Get image from stream
    Syncfusion.Drawing.Image image =
        Syncfusion.Drawing.Image.FromStream(imageStream);
    ///Add ole object
    IOleObject oleObject = worksheet.OleObjects.Add(inputStream, image,
        OleObjectType.PowerPointPresentation);
    ///Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    ///Save the document as file and view the saved document
    ///The operation in SaveAndView under Xamarin varies between Windows Phone,
///Android, and iOS platforms. Refer to the xlsio/xamarin section for
///respective code samples
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Embedd
edObjects.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("EmbeddedObjects.xl
sx", "application/msexcel", stream);
    }
}

```

The following code example illustrates how to insert and manipulate OLEObjects with their properties.

To learn more about OLEObjects, refer to the **IOleObjects** in API section.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    ///Icon image
    Image image = Image.FromFile("Images.png");
}

```

```

//Select the object, image for the display icon and the type of the
OleObject to insert
IOleObject oleObject1 = sheet.OleObjects.Add("Employee.docx", image,
OleLinkType.Embed);
//Select the object, image for the display icon and the type of the
OleObject to insert
IOleObject oleObject2 = sheet.OleObjects.Add("Customer.docx", image,
OleLinkType.Embed);
//Displays image as icon
oleObject1.DisplayAsIcon = true;
//Set the location of the OleObject
oleObject1.Location = sheet["K8"];
//Reads icon as a image
Image image1 = oleObject1.Picture;
//Reads OleObject as a IPictureShape
IPictureShape shape = oleObject1.Shape;
//Set the size of the OleObject
oleObject1.Size = new Size(30, 30);
//Remove OleObjects
oleObject2 = sheet.OleObjects[1];
oleObject2.Shape.Remove();
workbook.SaveAs("OleObjects.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Icon image
Dim image As Image = Image.FromFile("Images.png")
'Select the object, image for the display icon and the type of the OleObject
to insert
Dim oleObject1 As IOleObject = sheet.OleObjects.Add("Employee.docx", image,
OleLinkType.Embed)
'Select the object, image for the display icon and the type of the OleObject
to insert
Dim oleObject2 As IOleObject = sheet.OleObjects.Add("Customer.docx", image,
OleLinkType.Embed)
'Displays image as icon
oleObject1.DisplayAsIcon = True
'Set the location of the OleObject
oleObject1.Location = sheet("K8")
'Reads icon as a image
Dim image1 As Image = oleObject1.Picture
'Reads OleObject as a IPictureShape
Dim shape As IPictureShape = oleObject1.Shape
'Set the size of the OleObject
oleObject1.Size = New Size(30, 30)
'Remove OleObjects
oleObject2 = sheet.OleObjects(1)
oleObject2.Shape.Remove()
workbook.SaveAs("OleObjects.xlsx")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    // "App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream1 =
        assembly.GetManifestResourceStream("UWP.Data.Employee.docx");
    Stream inputStream2 =
        assembly.GetManifestResourceStream("UWP.Data.Customer.docx");
    Stream imageStream =
        assembly.GetManifestResourceStream("UWP.Data.Images.png");
    // Get image from stream
    Syncfusion.XlsIO.Image image =
        Syncfusion.XlsIO.Image.FromStream(imageStream);
    // Add ole object
    IOleObject oleObject1 = worksheet.OleObjects.Add(inputStream1, image,
        OleObjectType.WordDocument);
    IOleObject oleObject2 = worksheet.OleObjects.Add(inputStream2, image,
        OleObjectType.WordDocument);
    // Displays image as icon
    oleObject1.DisplayAsIcon = true;
    // Set the location of the OleObject
    oleObject1.Location = worksheet["K8"];
    // Reads icon as a image
    Syncfusion.XlsIO.Image image1 = oleObject1.Picture;
    // Reads OleObject as a IPictureShape
    IPictureShape shape = oleObject1.Shape;
    // Set the size of the OleObject
    oleObject1.Size = new Size(30, 30);
    // Remove OleObjects
    oleObject2 = worksheet.OleObjects[1];
    oleObject2.Shape.Remove();
    // Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "OleObjects";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    // Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    // Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;

```

```

IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Create file stream and image stream
FileStream inputStream1 = new FileStream("Employee.docx", FileMode.Open);
FileStream inputStream2 = new FileStream("Customer.docx", FileMode.Open);
FileStream imageStream = new FileStream("Images.png", FileMode.Open);
//Get image from stream
Image image = Image.FromStream(imageStream);
//Add ole object
IOleObject oleObject1 = worksheet.OleObjects.Add(inputStream1, image,
OleObjectType.WordDocument);
IOleObject oleObject2 = worksheet.OleObjects.Add(inputStream2, image,
OleObjectType.WordDocument);
//Displays image as icon
oleObject1.DisplayAsIcon = true;
//Set the location of the OleObject
oleObject1.Location = worksheet["K8"];
//Reads icon as a image
Image image1 = oleObject1.Picture;
//Reads OleObject as a IPictureShape
IPictureShape shape = oleObject1.Shape;
//Set the size of the OleObject
oleObject1.Size = new Size(30, 30);
//Remove OleObjects
oleObject2 = worksheet.OleObjects[1];
oleObject2.Shape.Remove();
//Saving the workbook as stream
FileStream stream = new FileStream("OleObjects.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    // "App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream1 =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Emp
        loyee.docx");
    Stream inputStream2 =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Cus
        tomer.docx");
    Stream imageStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Ima
        ges.png");
    //Get image from stream
    Syncfusion.Drawing.Image image =
        Syncfusion.Drawing.Image.FromStream(imageStream);
    //Add ole object

```

```

IOleObject oleObject1 = worksheet.OleObjects.Add(inputStream1, image,
OleObjectType.WordDocument);
IOleObject oleObject2 = worksheet.OleObjects.Add(inputStream2, image,
OleObjectType.WordDocument);
//Displays image as icon
oleObject1.DisplayAsIcon = true;
//Set the location of the OleObject
oleObject1.Location = worksheet["K8"];
//Reads icon as a image
Syncfusion.Drawing.Image image1 = oleObject1.Picture;
//Reads OleObject as a IPictureShape
IPictureShape shape = oleObject1.Shape;
//Set the size of the OleObject
oleObject1.Size = new Syncfusion.Drawing.Size(30, 30);
//Remove OleObjects
oleObject2 = worksheet.OleObjects[1];
oleObject2.Shape.Remove();
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
Android, and iOS platforms. Refer to the xlsio/xamarin section for
respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("OleObj
ects.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("OleObjects.xlsx",
"application/msexcel", stream);
}
}

```

Working with Macros

Macro is a set of process that can be run repeatedly in Excel document.

Creating a Macro

XlsIO allows to create macros in Excel document through **IVbaProject** interface. The macro document can be saved into different formats such as XLS, XLTM and XLSM.

XlsIO supports creating macro in following types using **VbaModuleType** enum.

- Document
- StdModule
- Class
- MsForm

You can add a Vba module through **IVbaModules** interface in XlsIO.

Document

Document is the default module type which will be added for every worksheet and one for entire workbook while creating VbaProject.

C#

```
//Adding Document to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("Document", VbaModuleType.Document);
```

VB.NET

```
//Adding Document to the workbook
Dim project As IVbaProject = workbook.VbaProject
Dim [module] As IVbaModule = project.Modules.Add("Document ", VbaModuleType.Document)
```

UWP

```
//Adding Document to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("Document ", VbaModuleType.Document);
```

ASP.NET CORE

```
//Adding Document to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("Document ", VbaModuleType.Document);
```

XAMARIN

```
//Adding Document to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("Document", VbaModuleType.Document);
```

The following code illustrate how to use Document module in Excel document.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Instantiate the excel application object.
    IApplication application = excelEngine.Excel;
    // Creating new workbook
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Accessing sheet module
    IVbaModule vbaModule = vbaModules[sheet.CodeName];
```

```
//Adding vba code to the module
vbaModule.Code = "Sub Auto_Open\n MsgBox \"Workbook Opened\" \n End Sub";
//Saving as macro document
workbook.SaveAs("Output.xlsm");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
'Instantiate the excel application object.
Dim application As IApplication = excelEngine.Excel
'Creating new Workbook
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creating Vba project
Dim project As IVbaProject = workbook.VbaProject
'Accessing vba modules collection
Dim vbaModules As IVbaModules = project.Modules
'Accessing sheet module
Dim vbaModule As IVbaModule = vbaModules(sheet.CodeName)
'Adding vba code to the module
vbaModule.Code = "Sub Auto_Open" & vbLf & " MsgBox "" Workbook Opened "" " &
vbLf & " End Sub"
'Saving as macro document
workbook.SaveAs("Output.xlsm")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Creating new workbook
IWorkbook workbook = application.Workbooks.Create(1);
workbook.Version = ExcelVersion.Excel2016;
IWorksheet worksheet = workbook.Worksheets[0];
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
// Accessing sheet module
IVbaModule vbaModule = vbaModules[sheet.CodeName];
//Adding vba code to the module
vbaModule.Code = "Sub Auto_Open\n MsgBox \" Workbook Opened \" \n End Sub";
// Save the Workbook
StorageFile storageFile;
if
(! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons")))
{
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsm"
});
storageFile = await savePicker.PickSaveFileAsync();
}
```



```

}
else
{
StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
storageFile = await local.CreateFileAsync("Output.xlsm",
CreationCollisionOption.ReplaceExisting);
}
//Saving the workbook
await workbook.SaveAsAsync(storageFile, ExcelSaveType.SaveAsMacro);
// Launch the saved file
await Windows.System.Launcher.LaunchFileAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
//Instantiate the excel application object.
IApplication application = excelEngine.Excel;
// Creating new workbook
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
// Accessing sheet module
IVbaModule vbaModule = vbaModules[sheet.CodeName];
//Adding vba code to the module
vbaModule.Code = "Sub Auto_Open\n MsgBox \" Workbook Opened \" \n End Sub";
//Saving as macro document
FileStream output = new FileStream("Output.xlsm", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(output, ExcelSaveType.SaveAsMacro);
}

```

XAMARIN

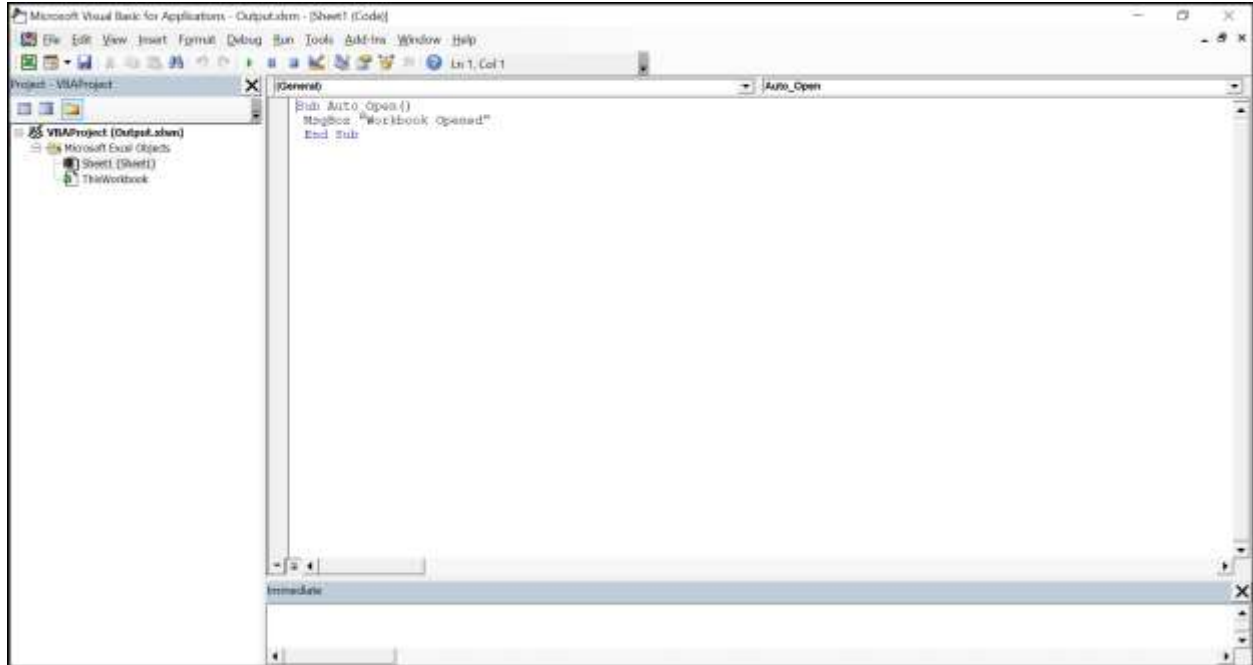
```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Creating new workbook
IWorkbook workbook = application.Workbooks.Create(1);
workbook.Version = ExcelVersion.Excel2016;
IWorksheet worksheet = workbook.Worksheets[0];
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Accessing sheet module
IVbaModule vbaModule = vbaModules[sheet.CodeName];
//Adding vba code to the module
vbaModule.Code = "Sub Auto_Open\n MsgBox \"Workbook Opened\" \n End Sub";
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream, ExcelSaveType.SaveAsMacro);
//Save the stream into XLSM file

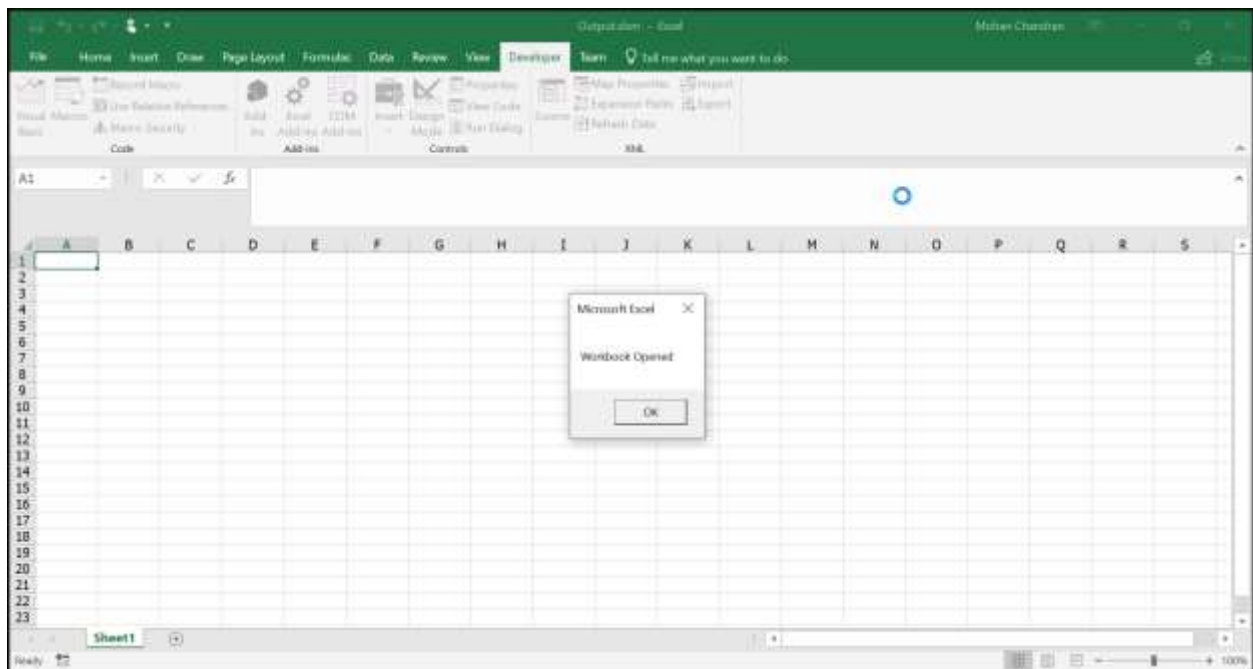
```

```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("sample.xlsx",  
"application/msexcel", stream);  
}
```

The Vba project in the output looks like below.



The macro output in the Excel document looks like below.



[StdModule](#)

StdModule is the module created for whenever a macro process is recorded in Excel document.

The following code illustrate how to add a StdModule using Add method. Here, the parameters name and **VbaModuleType** enum value is used.

- Test – Macro module name added to the Vba project.
- StdModule – Type of the Vba module.

C#

```
//Adding StdModule to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("Test", VbaModuleType.StdModule);
```

VB.NET

```
//Adding StdModule to the workbook
Dim project As IVbaProject = workbook.VbaProject
Dim [module] As IVbaModule = project.Modules.Add("Test",
VbaModuleType.StdModule)
```

UWP

```
//Adding StdModule to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("Test", VbaModuleType.StdModule);
```

ASP.NET CORE

```
//Adding StdModule to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("Test", VbaModuleType.StdModule);
```

XAMARIN

```
//Adding StdModule to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("Test", VbaModuleType.StdModule);
```

The following code illustrate how to create a macro using StdModule in Excel document.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Instantiate the excel application object.
    IApplication application = excelEngine.Excel;
    // Creating new workbook
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Adding a vba module
```

```

IVbaModule vbaModule = vbaModules.Add("StdModule", VbaModuleType.StdModule);
//Adding vba code to the module
vbaModule.Code = "Sub Auto_Open\n MsgBox \"Macro Added\" \n End Sub";
//Saving as macro document
workbook.SaveAs("Output.xlsm");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
    'Instantiate the excel application object.
    Dim application As IApplication = excelEngine.Excel
    'Creating new Workbook
    Dim workbook As IWorkbook = application.Workbooks.Create(1)
    Dim sheet As IWorksheet = workbook.Worksheets(0)
    'Creating Vba project
    Dim project As IVbaProject = workbook.VbaProject
    'Accessing vba modules collection
    Dim vbaModules As IVbaModules = project.Modules
    'Adding a vba module
    Dim vbaModule As IVbaModule = vbaModules.Add("StdModule",
        VbaModuleType.StdModule)
    'Adding vba code to the module
    vbaModule.Code = "Sub Auto_Open" & vbLf & " MsgBox \"Macro Added\" " & vbLf
    & " End Sub"
    'Saving as macro document
    workbook.SaveAs("Output.xlsm")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Creating new workbook
    IWorkbook workbook = application.Workbooks.Create(1);
    workbook.Version = ExcelVersion.Excel2016;
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Adding a vba module
    IVbaModule vbaModule = vbaModules.Add("StdModule", VbaModuleType.StdModule);
    //Adding vba code to the module
    vbaModule.Code = "Sub Auto_Open\n MsgBox \"Macro Added\" \n End Sub";
    // Save the Workbook
    StorageFile storageFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
        savePicker.SuggestedFileName = "Output";
    }
}

```

```

savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsm"
});
storageFile = await savePicker.PickSaveFileAsync();
}
else
{
StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
storageFile = await local.CreateFileAsync("Output.xlsm",
CreationCollisionOption.ReplaceExisting);
}
//Saving the workbook
await workbook.SaveAsAsync(storageFile, ExcelSaveType.SaveAsMacro);
// Launch the saved file
await Windows.System.Launcher.LaunchFileAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
//Instantiate the excel application object.
IApplication application = excelEngine.Excel;
// Creating new workbook
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Adding a vba module
IVbaModule vbaModule = vbaModules.Add("StdModule", VbaModuleType.StdModule);
//Adding vba code to the module
vbaModule.Code = "Sub Auto_Open\n MsgBox \"Macro Added\" \n End Sub";
//Saving as macro document
FileStream output = new FileStream("Output.xlsm", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(output, ExcelSaveType.SaveAsMacro);
}

```

XAMARIN

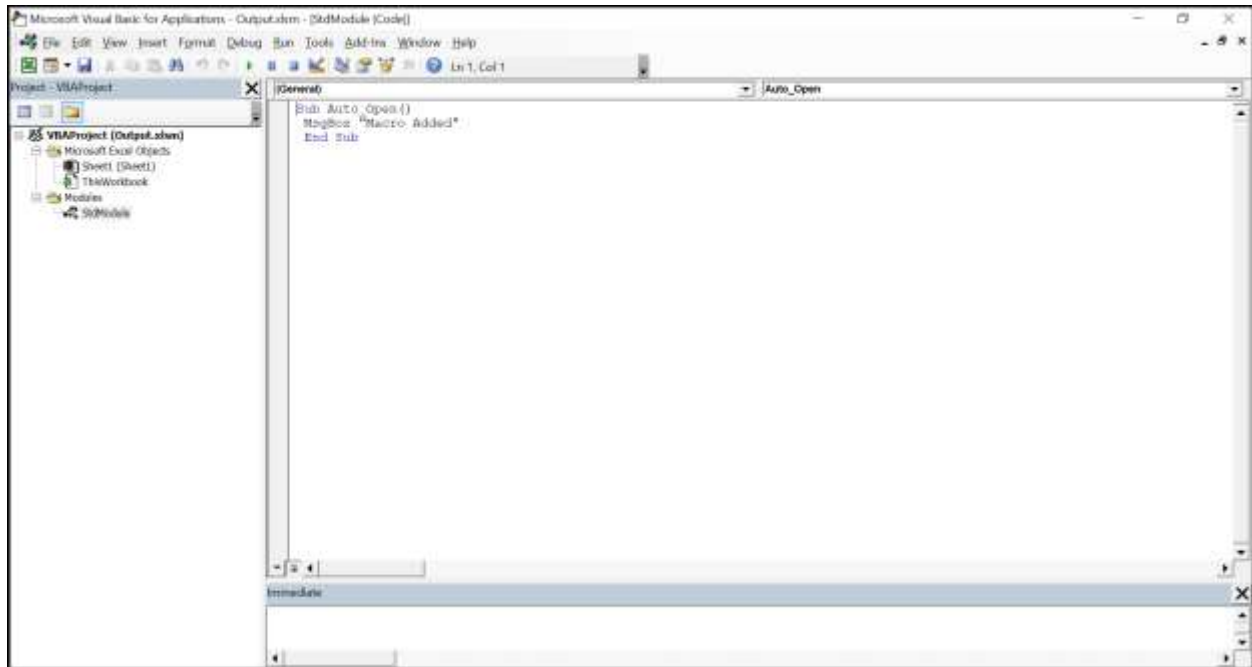
```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Creating new workbook
IWorkbook workbook = application.Workbooks.Create(1);
workbook.Version = ExcelVersion.Excel2016;
IWorksheet worksheet = workbook.Worksheets[0];
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Adding a vba module
IVbaModule vbaModule = vbaModules.Add("StdModule", VbaModuleType.StdModule);
//Adding vba code to the module
vbaModule.Code = "Sub Auto_Open\n MsgBox \"Macro Added\" \n End Sub";
}

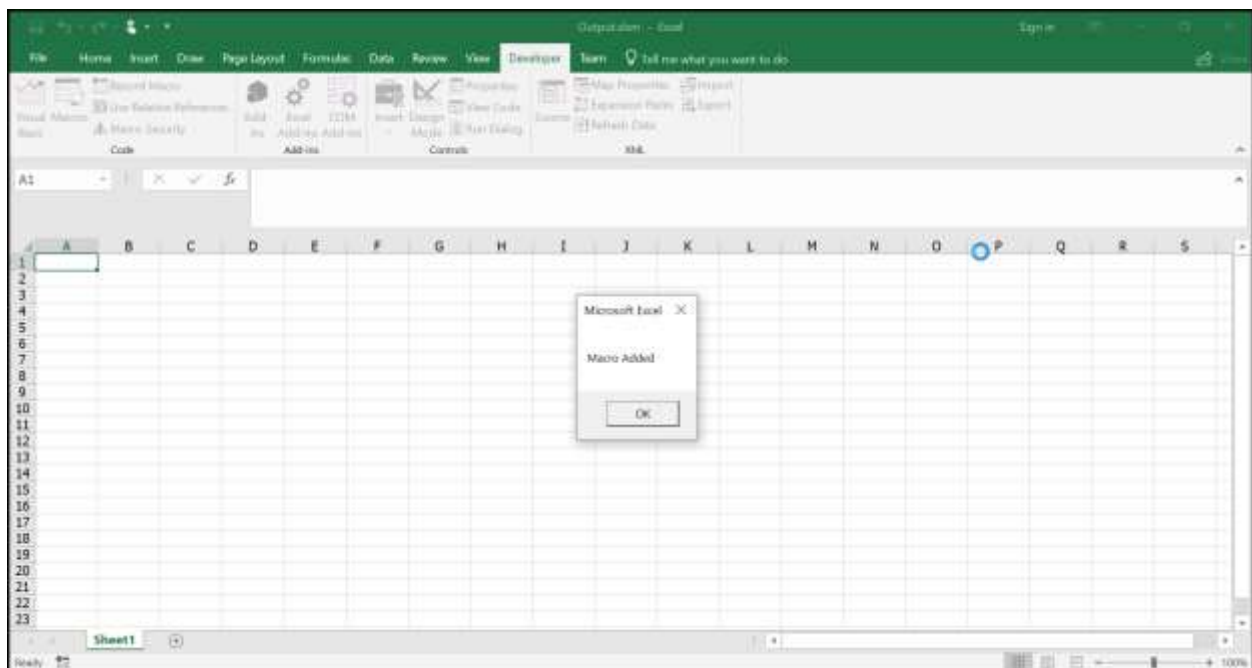
```

```
MemoryStream stream = new MemoryStream();  
workbook.SaveAs(stream, ExcelSaveType.SaveAsMacro);  
//Save the stream into XLSX file  
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("sample.xlsx",  
"application/msexcel", stream);  
}
```

The Vba project in the output Excel document looks like below.



The Macro output in the Excel document looks like below.



Class

Class module allows us to create our own object model to use it where same kind of objects needs to be added with different values such as creating the employee information list. XlsIO supports creating a class module in Excel document.

The following code illustrate how to add a class in XlsIO. Here, the parameters name and VbaModuleType enum value is used.

- Test – Class name used in the Vba project
- ClassModule – Type of Vba module

C#

```
//Adding class module to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("Test", VbaModuleType.ClassModule);
```

VB.NET

```
//Adding class module to the workbook
Dim project As IVbaProject = workbook.VbaProject
Dim [module] As IVbaModule = project.Modules.Add("Test",
VbaModuleType.ClassModule)
```

UWP

```
//Adding class module to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("Test", VbaModuleType.ClassModule);
```

ASP.NET CORE

```
//Adding class module to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("Test", VbaModuleType.ClassModule);
```

XAMARIN

```
//Adding class module to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("Test", VbaModuleType.StdModule);
```

The following code illustrate how to use class module to run a macro with another module in Excel document.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Instantiate the excel application object.
    IApplication application = excelEngine.Excel;
    // Creating new workbook
    IWorkbook workbook = application.Workbooks.Create(1);
```

```

IWorksheet sheet = workbook.Worksheets[0];
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Adding a class module
IVbaModule clsModule = vbaModules.Add("Test", VbaModuleType.ClassModule);
clsModule.Code = "Public Sub Create()\n MsgBox \"Created a class module\" \n\n End Sub";
//Adding a vba module
IVbaModule vbaModule = vbaModules.Add("Module1", VbaModuleType.StdModule);
//Using class in StdModule
vbaModule.Code = "Sub Auto_Open()\n Dim obj As New test \n obj.Create \n End Sub";
//Saving as macro document
workbook.SaveAs("Output.xlsm");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
'Instantiate the excel application object.
Dim application As IApplication = excelEngine.Excel
'Creating new Workbook
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creating Vba project
Dim project As IVbaProject = workbook.VbaProject
'Accessing vba modules collection
Dim vbaModules As IVbaModules = project.Modules
'Adding a vba module
Dim vbaModule As IVbaModule = vbaModules.Add("StdModule",
VbaModuleType.StdModule)
'Adding vba code to the module
vbaModule.Code = "Sub Auto_Open" & vbCrLf & " MsgBox \"Macro Added\" " & vbCrLf
& " End Sub"
'Saving as macro document
workbook.SaveAs("Output.xlsm")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Creating new workbook
IWorkbook workbook = application.Workbooks.Create(1);
workbook.Version = ExcelVersion.Excel2016;
IWorksheet worksheet = workbook.Worksheets[0];
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Adding a class module
IVbaModule clsModule = vbaModules.Add("Test", VbaModuleType.ClassModule);

```



```

clsModule.Code = "Public Sub Create()\n MsgBox \"Created a class module\" \n
End Sub";
//Adding a vba module
IVbaModule vbaModule = vbaModules.Add("Module1", VbaModuleType.StdModule);
//Using class in StdModule
vbaModule.Code = "Sub Auto_Open()\n Dim obj As New test \n obj.Create \n End
Sub";
// Save the Workbook
StorageFile storageFile;
if
(! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent ("Windows.Phone.U
I.Input.HardwareButtons")))
{
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsm"
});
storageFile = await savePicker.PickSaveFileAsync();
}
else
{
StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
storageFile = await local.CreateFileAsync("Output.xlsm",
CreationCollisionOption.ReplaceExisting);
}
//Saving the workbook
await workbook.SaveAsAsync(storageFile, ExcelSaveType.SaveAsMacro);
// Launch the saved file
await Windows.System.Launcher.LaunchFileAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
//Instantiate the excel application object.
IApplication application = excelEngine.Excel;
// Creating new workbook
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Adding a class module
IVbaModule clsModule = vbaModules.Add("Test", VbaModuleType.ClassModule);
clsModule.Code = "Public Sub Create()\n MsgBox \"Created a class module\" \n
End Sub";
//Adding a vba module
IVbaModule vbaModule = vbaModules.Add("Module1", VbaModuleType.StdModule);
//Using class in StdModule
vbaModule.Code = "Sub Auto_Open()\n Dim obj As New test \n obj.Create \n End
Sub";
//Saving as macro document

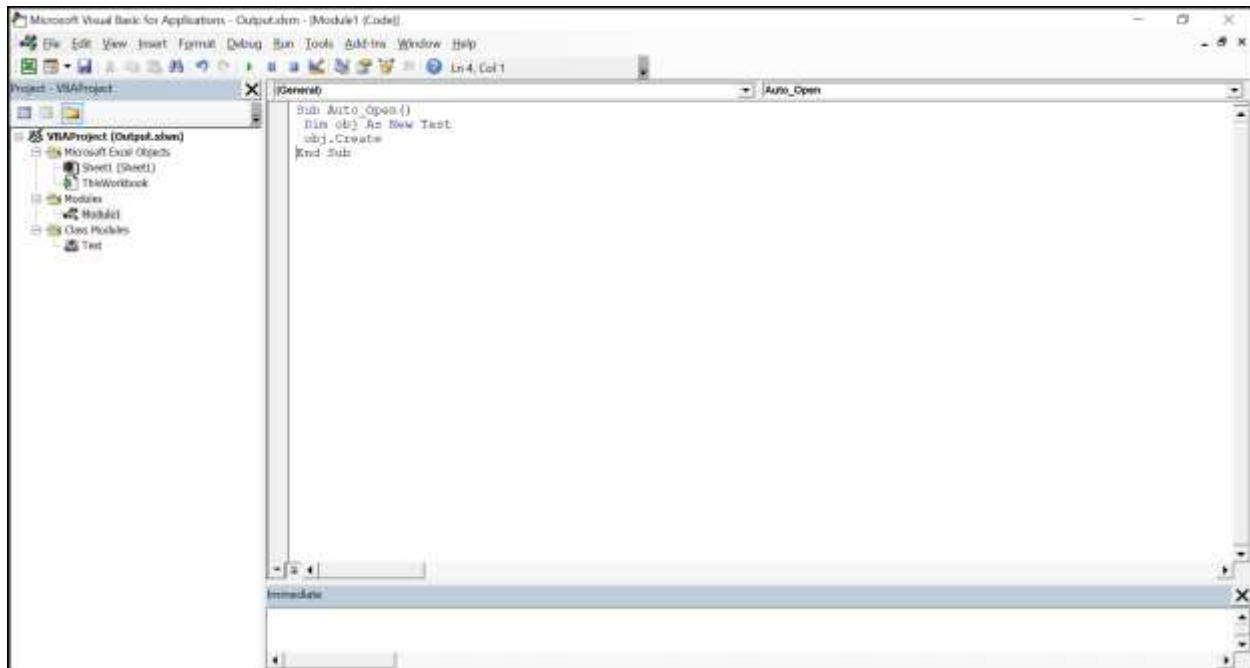
```

```
FileStream output = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(output, ExcelSaveType.SaveAsMacro);
}
```

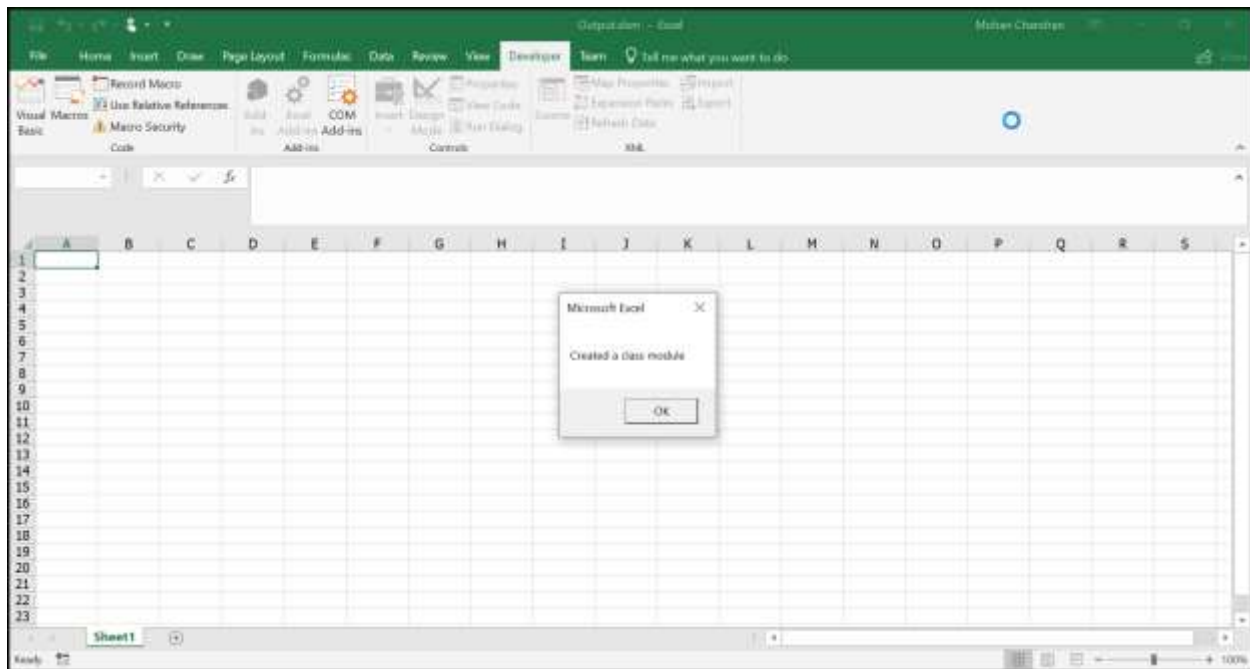
XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Creating new workbook
    IWorkbook workbook = application.Workbooks.Create(1);
    workbook.Version = ExcelVersion.Excel2016;
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Adding a class module
    IVbaModule clsModule = vbaModules.Add("Test", VbaModuleType.ClassModule);
    clsModule.Code = "Public Sub Create()\n MsgBox \"Created a class module\" \n\n End Sub";
    //Adding a vba module
    IVbaModule vbaModule = vbaModules.Add("Module1", VbaModuleType.StdModule);
    //Using class in StdModule
    vbaModule.Code = "Sub Auto_Open()\n Dim obj As New test \n obj.Create \n End Sub";
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream, ExcelSaveType.SaveAsMacro);
    //Save the stream into XLSX file
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("sample.xlsm",
    "application/msexcel", stream);
}
```

The Vba project in the output Excel document looks like below.



The Macro output in the Excel document looks like below.



MsForm

MsForm is the form module in which we can have form controls such as textbox, label, buttons etc. Form module cannot be created by XlsIO, but it allows to copy from a form module existing another workbook to new workbook.

The following code illustrate how to add a class in XlsIO. Here, the parameters name and VbaModuleType enum value is used.

- UserForm – Class name used in the Vba project

- MsForm – Type of Vba module

C#

```
//Adding class module to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("UserForm", VbaModuleType.MsForm);
```

VB.NET

```
//Adding class module to the workbook
Dim project As IVbaProject = workbook.VbaProject
Dim [module] As IVbaModule = project.Modules.Add("UserForm",
VbaModuleType.MsForm)
```

UWP

```
//Adding class module to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("UserForm", VbaModuleType.MsForm);
```

ASP.NET CORE

```
//Adding class module to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("UserForm", VbaModuleType.MsForm);
```

XAMARIN

```
//Adding class module to the workbook
IVbaProject project = workbook.VbaProject;
IVbaModule module = project.Modules.Add("UserForm", VbaModuleType.MsForm);
```

The following code illustrate how to copy a form from another workbook to new workbook.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Instantiate the excel application object.
    IApplication application = excelEngine.Excel;
    // Creating new workbook
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Opening form module existing workbook
    IWorkbook newBook = application.Workbooks.Open("Test.xls");
    IVbaProject newProject = newBook.VbaProject;
    //Accessing existing form module
    IVbaModule form = newProject.Modules["UserForm1"];
    //Adding a form module in new workbook
```

```

IVbaModule formModule = project.Modules.Add(form.Name,
VbaModuleType.MsForm);
//Copying the form code behind
formModule.Code = form.Code;
//Copying the designer of the form
formModule.DesignerStorage = form.DesignerStorage;
//Saving as macro document
workbook.SaveAs ("Output.xlsm");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
'Instantiate the excel application object.
Dim application As IApplication = excelEngine.Excel
'Creating new Workbook
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creating Vba project
Dim project As IVbaProject = workbook.VbaProject
'Accessing vba modules collection
Dim vbaModules As IVbaModules = project.Modules
'Open form existing workbook
Dim newBook As IWorkbook = application.Workbooks.Open("Test.xls")
Dim newProject As IVbaProject = newBook.VbaProject
'Accessing existing form module
Dim form As IVbaModule = newProject.Modules("UserForm1")
'Adding a form module in new workbook
Dim formModule As IVbaModule = project.Modules.Add(form.Name,
VbaModuleType.MsForm)
'Copying the form code behind
formModule.Code = form.Code
'Copying the designer of the form
formModule.DesignerStorage = form.DesignerStorage
'Saving as macro document
workbook.SaveAs ("Output.xlsm")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Creating new workbook
IWorkbook workbook = application.Workbooks.Create(1);
workbook.Version = ExcelVersion.Excel2016;
IWorksheet worksheet = workbook.Worksheets[0];
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsm");
openPicker.FileTypeFilter.Add(".xltm");
}

```

```

openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Open form existing workbook
IWorkbook newBook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
IVbaProject newProject = newBook.VbaProject;
//Accessing existing form module
IVbaModule form = newProject.Modules["UserForm1"];
//Adding a form module in new workbook
IVbaModule formModule = project.Modules.Add(form.Name,
VbaModuleType.MsForm);
//Copying the form code behind
formModule.Code = form.Code;
//Copying the designer of the form
formModule.DesignerStorage = form.DesignerStorage;
// Save the Workbook
StorageFile storageFile;
if
(! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
{
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsm"
});
storageFile = await savePicker.PickSaveFileAsync();
}
else
{
StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
storageFile = await local.CreateFileAsync("Output.xlsm",
CreationCollisionOption.ReplaceExisting);
}
//Saving the workbook
await workbook.SaveAsAsync(storageFile, ExcelSaveType.SaveAsMacro);
// Launch the saved file
await Windows.System.Launcher.LaunchFileAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
//Instantiate the excel application object.
IApplication application = excelEngine.Excel;
// Creating new workbook
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Opening form module existing workbook
FileStream input = new FileStream(DataPathBase + "Test.xls", FileMode.Open,
FileAccess.ReadWrite);

```

```

IWorkbook newBook = application.Workbooks.Open(input);
IVbaProject newProject = newBook.VbaProject;
//Accessing existing form module
IVbaModule form = newProject.Modules["UserForm1"];
//Adding a form module in new workbook
IVbaModule formModule = project.Modules.Add(form.Name,
VbaModuleType.MsForm);
//Copying the form code behind
formModule.Code = form.Code;
//Copying the designer of the form
formModule.DesignerStorage = form.DesignerStorage;
//Saving as macro document
FileStream output = new FileStream("Output.xlsm", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(output, ExcelSaveType.SaveAsMacro);
}

```

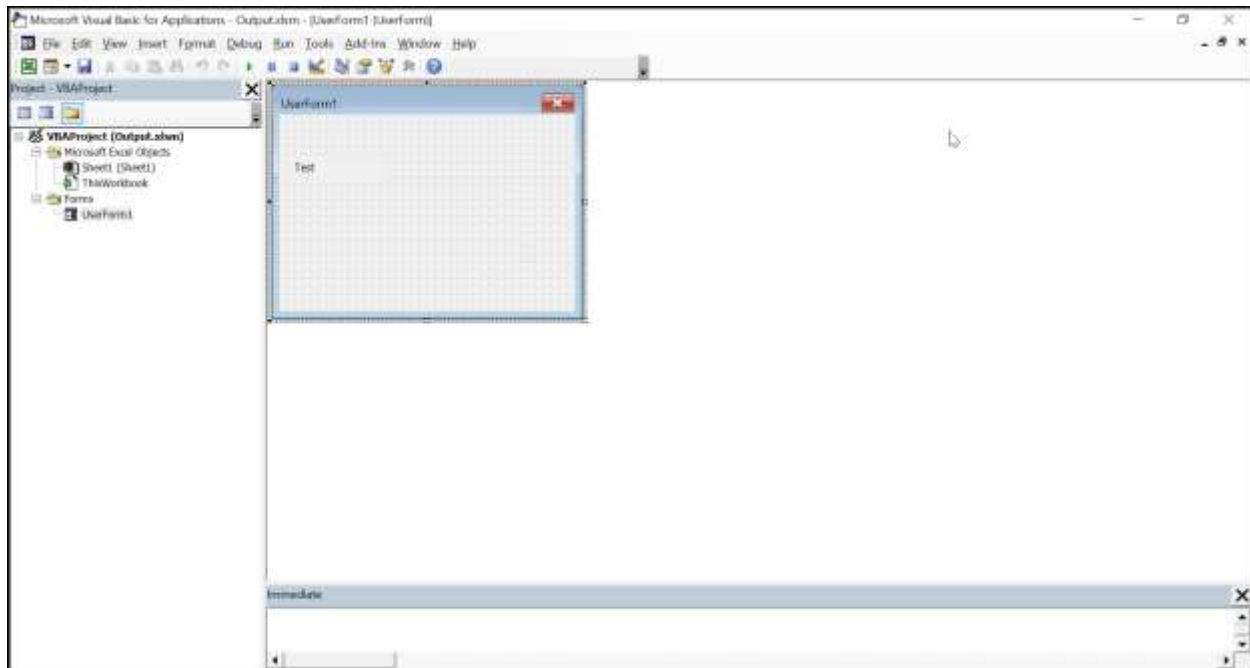
XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Creating new workbook
IWorkbook workbook = application.Workbooks.Create(1);
workbook.Version = ExcelVersion.Excel2016;
IWorksheet worksheet = workbook.Worksheets[0];
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//"App2" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream("App.Test.xls");
//Opening form module existing workbook
IWorkbook newBook = excelEngine.Excel.Workbooks.Open(inputStream);
IVbaProject newProject = newBook.VbaProject;
//Accessing existing form module
IVbaModule form = newProject.Modules["UserForm1"];
//Adding a form module in new workbook
IVbaModule formModule = project.Modules.Add(form.Name,
VbaModuleType.MsForm);
//Copying the form code behind
formModule.Code = form.Code;
//Copying the designer of the form
formModule.DesignerStorage = form.DesignerStorage;
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream, ExcelSaveType.SaveAsMacro);
//Save the stream into XLSX file
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("sample.xlsm",
"application/msexcel", stream);
}

```

The Vba project in the output Excel document looks like below.



Assigning Macro to Shapes

XlsIO supports assigning macros to the shape controls in the Excel document through **OnAction** property.

The following code illustrate how to assign macros to shapes in Excel document.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Instantiate the excel application object.
    IApplication application = excelEngine.Excel;
    // Creating new workbook
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Adding a vba module
    IVbaModule vbaModule = vbaModules.Add("StdModule", VbaModuleType.StdModule);
    //Adding vba code to the module
    vbaModule.Code = "Sub Invoke()\n MsgBox \"Macro Added\" \n End Sub";
    //Adding a auto shape
    IShape shape = sheet.Shapes.AddAutoShapes(AutoShapeType.Rectangle, 1, 2, 60,
    70);
    shape.Name = "Shape1";
    //Assigning a Macro to shape
    shape.OnAction = "StdModule.Invoke";
    //Saving as macro document
    workbook.SaveAs("Output.xlsm");
}
```

VB.NET


```

Using excelEngine As ExcelEngine = New ExcelEngine()
'Instantiate the excel application object.
Dim application As IApplication = excelEngine.Excel
'Creating new Workbook
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Creating Vba project
Dim project As IVbaProject = workbook.VbaProject
'Accessing vba modules collection
Dim vbaModules As IVbaModules = project.Modules
'Adding a vba module
Dim vbaModule As IVbaModule = vbaModules.Add("StdModule",
VbaModuleType.StdModule)
'Adding vba code to the module
vbaModule.Code = "Sub Invoke()" & vbLf & " MsgBox ""Macro Added"" " & vbLf &
" End Sub"
'Adding a auto shape
Dim shape As IShape = sheet.Shapes.AddAutoShapes(AutoShapeType.Rectangle, 1,
2, 60, 70)
shape.Name = "Shape1"
'Assigning a Macro to shape
shape.OnAction = "StdModule.Invoke"
'Saving as macro document
workbook.SaveAs("Output.xlsm")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Creating new workbook
IWorkbook workbook = application.Workbooks.Create(1);
workbook.Version = ExcelVersion.Excel2016;
IWorksheet worksheet = workbook.Worksheets[0];
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Adding a vba module
IVbaModule vbaModule = vbaModules.Add("StdModule", VbaModuleType.StdModule);
//Adding vba code to the module
vbaModule.Code = "Sub Invoke()\n MsgBox \"Macro Added\" \n End Sub";
//Adding a auto shape
IShape shape = sheet.Shapes.AddAutoShapes(AutoShapeType.Rectangle, 1, 2, 60,
70);
shape.Name = "Shape1";
//Assigning a Macro to shape
shape.OnAction = "StdModule.Invoke";
// Save the Workbook
StorageFile storageFile;
if
(! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
{
FileSavePicker savePicker = new FileSavePicker();

```

```

savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsm"
});
storageFile = await savePicker.PickSaveFileAsync();
}
else
{
StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
storageFile = await local.CreateFileAsync("Output.xlsm",
CreationCollisionOption.ReplaceExisting);
}
//Saving the workbook
await workbook.SaveAsAsync(storageFile, ExcelSaveType.SaveAsMacro);
// Launch the saved file
await Windows.System.Launcher.LaunchFileAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
//Instantiate the excel application object.
IApplication application = excelEngine.Excel;
// Creating new workbook
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet sheet = workbook.Worksheets[0];
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Adding a vba module
IVbaModule vbaModule = vbaModules.Add("StdModule", VbaModuleType.StdModule);
//Adding vba code to the module
vbaModule.Code = "Sub Invoke()\n MsgBox \"Macro Added\" \n End Sub";
//Adding a auto shape
IShape shape = sheet.Shapes.AddAutoShapes(AutoShapeType.Rectangle, 1, 2, 60,
70);
shape.Name = "Shape1";
//Assigning a Macro to shape
shape.OnAction = "StdModule.Invoke";
//Saving as macro document
FileStream output = new FileStream("Output.xlsm", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(output, ExcelSaveType.SaveAsMacro);
}

```

XAMARIN

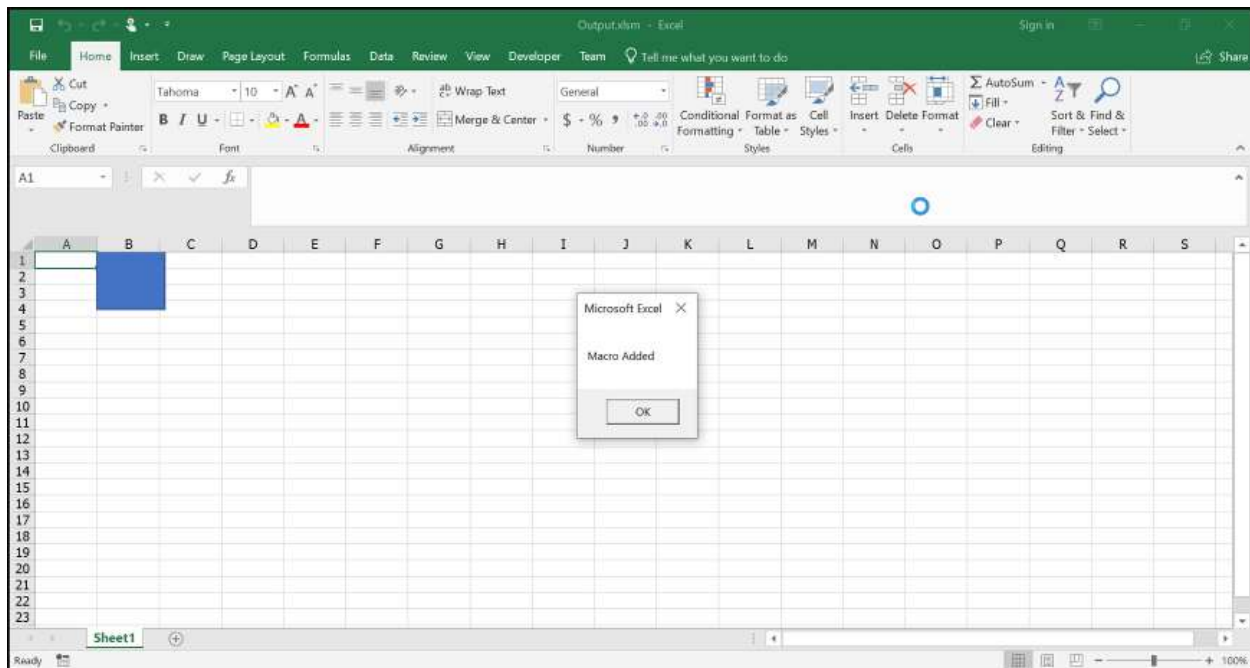
```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Creating new workbook
IWorkbook workbook = application.Workbooks.Create(1);
workbook.Version = ExcelVersion.Excel2016;
IWorksheet worksheet = workbook.Worksheets[0];

```

```
//Creating Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Adding a vba module
IVbaModule vbaModule = vbaModules.Add("StdModule", VbaModuleType.StdModule);
//Adding vba code to the module
vbaModule.Code = "Sub Invoke()\n MsgBox \"Macro Added\" \n End Sub";
//Adding a auto shape
IShape shape = sheet.Shapes.AddAutoShapes(AutoShapeType.Rectangle, 1, 2, 60,
70);
shape.Name = "Shape1";
//Assigning a Macro to shape
shape.OnAction = "StdModule.Invoke";
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream, ExcelSaveType.SaveAsMacro);
//Save the stream into XLSX file
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("sample.xlsm",
"application/msexcel", stream);
}
```

When the shape is clicked, the output looks like below.



Saving macro enabled document into stream

By default, while saving the Excel workbook into stream, the file type will be based on the Excel version used. For Excel97to2003 version, the file format will be XLS type. Above this version, the document will be saved as XLSX format. So, while saving the macro enabled documents into XLSM and XLTM formats into stream, the **ExcelSaveType** should be provided as **SaveAsMacro** and **SaveAsMacroTemplate**.

The following code illustrate how to save macro-enabled documents into stream.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
```

```

{
    //Instantiate the excel application object.
    IApplication application = excelEngine.Excel;
    // Creating new workbook
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Adding a vba module
    IVbaModule vbaModule = vbaModules.Add("StdModule", VbaModuleType.StdModule);
    //Adding vba code to the module
    vbaModule.Code = "Sub Auto_Open\n MsgBox \"Macro Added\" \n End Sub";
    //Saving as Macro in XLSM format
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream, ExcelSaveType.SaveAsMacro);
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
    'Instantiate the excel application object.
    Dim application As IApplication = excelEngine.Excel
    'Creating new Workbook
    Dim workbook As IWorkbook = application.Workbooks.Create(1)
    Dim sheet As IWorksheet = workbook.Worksheets(0)
    'Creating Vba project
    Dim project As IVbaProject = workbook.VbaProject
    'Accessing vba modules collection
    Dim vbaModules As IVbaModules = project.Modules
    'Adding a vba module
    Dim vbaModule As IVbaModule = vbaModules.Add("StdModule",
    VbaModuleType.StdModule)
    'Adding vba code to the module
    vbaModule.Code = "Sub Auto_Open" & vbCrLf & " MsgBox \"Macro Added\" " & vbCrLf
    & " End Sub"
    'Saving as Macro in XLSM format
    Dim stream As MemoryStream = New MemoryStream()
    workbook.SaveAs(stream, ExcelSaveType.SaveAsMacro)
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Creating new workbook
    IWorkbook workbook = application.Workbooks.Create(1);
    workbook.Version = ExcelVersion.Excel2016;
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Adding a vba module

```

```

IVbaModule vbaModule = vbaModules.Add("StdModule", VbaModuleType.StdModule);
//Adding vba code to the module
vbaModule.Code = "Sub Auto_Open\n MsgBox \"Macro Added\" \n End Sub";
//Saving as Macro in XLTM format
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream, ExcelSaveType.SaveAsMacroTemplate);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Instantiate the excel application object.
    IApplication application = excelEngine.Excel;
    // Creating new workbook
    IWorkbook workbook = application.Workbooks.Create(1);
    workbook.Version = ExcelVersion.Excel2016;
    IWorksheet sheet = workbook.Worksheets[0];
    //Creating Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Adding a vba module
    IVbaModule vbaModule = vbaModules.Add("StdModule", VbaModuleType.StdModule);
    //Adding vba code to the module
    vbaModule.Code = "Sub Auto_Open\n MsgBox \"Macro Added\" \n End Sub";
    //Saving as Macro in XLTM format
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream, ExcelSaveType.SaveAsMacroTemplate);
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Creating new workbook
    IWorkbook workbook = application.Workbooks.Create(1);
    workbook.Version = ExcelVersion.Excel2016;
    IWorksheet worksheet = workbook.Worksheets[0];
    //Creating Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Adding a vba module
    IVbaModule vbaModule = vbaModules.Add("StdModule", VbaModuleType.StdModule);
    //Adding vba code to the module
    vbaModule.Code = "Sub Auto_Open\n MsgBox \"Macro Added\" \n End Sub";
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream, ExcelSaveType.SaveAsMacro);
}

```

Editing a Macro

XlsIO allows to edit the existing macros in the Excel documents. To edit macros in Excel document, the module containing the macro code needs to be modified. By using the name of the module, it can be accessed and edited in XlsIO.

The following code illustrate how to edit existing macro in Excel document.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    // Instantiate the excel application object.
    IApplication application = excelEngine.Excel;
    // Opening a workbook
    IWorkbook workbook = application.Workbooks.Open("Test.xls");
    workbook.Version = ExcelVersion.Excel2016;
    IWorksheet sheet = workbook.Worksheets[0];
    //Accessing Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Access a Vba Module
    IVbaModule vbaModule = vbaModules["Module1"];
    //Edit the macro
    vbaModule.Name = "CreateData";
    vbaModule.Code = "Sub Auto_Open()\n MsgBox \"Macro is edited\" \n End Sub ";
    //Saving as macro document
    workbook.SaveAs("Output.xlsm");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
    'Instantiate the excel application object.
    Dim application As IApplication = excelEngine.Excel
    'Opening a Workbook
    Dim workbook As IWorkbook = application.Workbooks.Open("Test.xls")
    Dim sheet As IWorksheet = workbook.Worksheets(0)
    'Creating Vba project
    Dim project As IVbaProject = workbook.VbaProject
    'Accessing vba modules collection
    Dim vbaModules As IVbaModules = project.Modules
    'Accessing a vba module
    Dim vbaModule As IVbaModule = vbaModules("Module1")
    'Editing a module
    vbaModule.Name = "CreateData"
    vbaModule.Code = "Sub Auto_Open() " & vbLf & " MsgBox ""Macro is edited"" " &
vbLf & " End Sub "
    'Saving as macro document
    workbook.SaveAs("Output.xlsm")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
```

```

IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsm");
openPicker.FileTypeFilter.Add(".xltn");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opening a workbook with a worksheet
IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
workbook.Version = ExcelVersion.Excel2016;
IWorksheet worksheet = workbook.Worksheets[0];
//Accessing Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Access a Vba Module
IVbaModule vbaModule = vbaModules["Module1"];
//Edit the macro
vbaModule.Name = "CreateData";
vbaModule.Code = "Sub Auto_Open()\n MsgBox \"Macro is edited\" \n End Sub ";
// Save the Workbook
StorageFile storageFile;
if
(! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
{
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsm"
});
storageFile = await savePicker.PickSaveFileAsync();
}
else
{
StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
storageFile = await local.CreateFileAsync("Output.xlsm",
CreationCollisionOption.ReplaceExisting);
}
//Saving the workbook
await workbook.SaveAsAsync(storageFile, ExcelSaveType.SaveAsMacro);
// Launch the saved file
await Windows.System.Launcher.LaunchFileAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
//Instantiate the excel application object.
IApplication application = excelEngine.Excel;
//Opening form module existing workbook
FileStream input = new FileStream(DataPathBase + "Test.xls", FileMode.Open,
FileAccess.ReadWrite);

```

```

IWorkbook workbook = application.Workbooks.Open(input);
IWorksheet sheet = workbook.Worksheets[0];
//Accessing Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Access a Vba Module
IVbaModule vbaModule = vbaModules["Module1"];
//Edit the macro
vbaModule.Name = "CreateData";
vbaModule.Code = "Sub Auto_Open()\n MsgBox \"Macro is edited\" \n End Sub ";
//Saving as macro document
FileStream output = new FileStream("Output.xlsm", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(output, ExcelSaveType.SaveAsMacro);
input.Close();
output.Close();
}

```

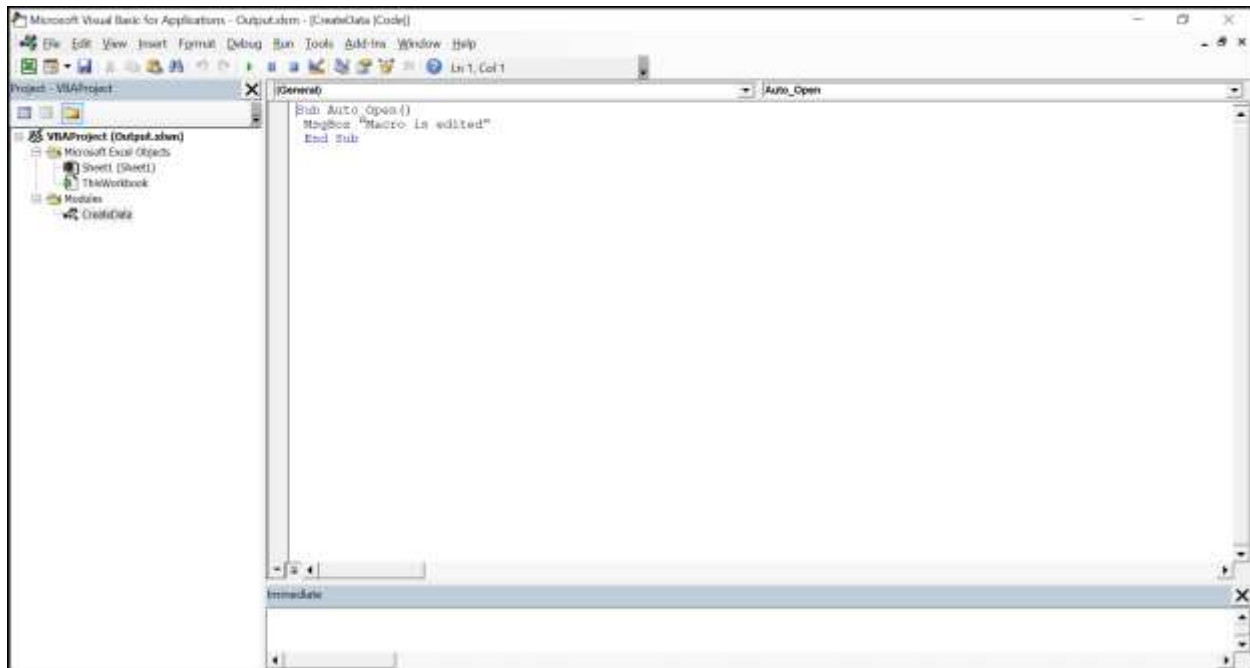
XAMARIN

```

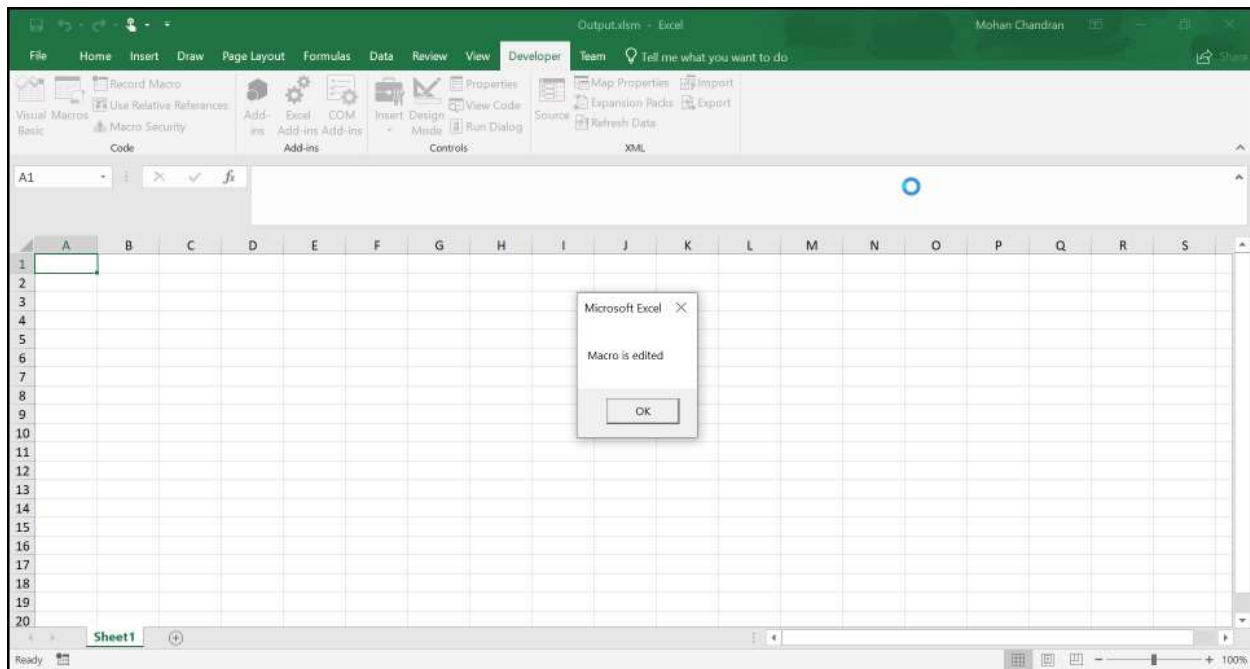
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream("App.Test.xls");
//Opening the workbook
IWorkbook workbook = application.Workbooks.Open(inputStream);
workbook.Version = ExcelVersion.Excel2016;
IWorksheet worksheet = workbook.Worksheets[0];
//Accessing Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Access a Vba Module
IVbaModule vbaModule = vbaModules["Module1"];
//Edit the macro
vbaModule.Name = "CreateData";
vbaModule.Code = "Sub Auto_Open()\n MsgBox \"Macro is edited\" \n End Sub ";
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream, ExcelSaveType.SaveAsMacro);
//Save the stream into XLSX file
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("sample.xlsm",
"application/msexcel", stream);
}

```

The Vba project in the output Excel document looks like below.



The Macro output in the Excel document looks like below.



Note: Macros are parsed only when accessed. By default, opening and saving a macro file will preserve its macros.

Removing Macros

Excel macro-enabled documents can be saved as normal documents where the macro process is not necessary. XlsIO supports saving the macro-enabled documents such as XLS, XLSM, XLTM without macros and normal documents such as XLSX and XLS. For that, macro in the Excel documents needs to be removed.

Macro in the Excel document can be removed in the following ways.

- Remove(String name)
- RemoveAt(int index)
- Clear()
- SkipOnSave

Remove(string name)

Macro process exist in the Vba project's code modules. To remove a macro, the Vba modules needs to be removed.

- name – Name of the Vba module needs to be removed.

The following code illustrate how to remove a module using Remove method.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    // Instantiate the excel application object.
    IApplication application = excelEngine.Excel;
    // Opening a workbook
    IWorkbook workbook = application.Workbooks.Open("Test.xls");
    workbook.Version = ExcelVersion.Excel2016;
    IWorksheet sheet = workbook.Worksheets[0];
    //Accessing Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Remove macro module
    vbaModules.Remove("Module1");
    //Saving as macro document
    workbook.SaveAs("Output.xlsm");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
    'Instantiate the excel application object.
    Dim application As IApplication = excelEngine.Excel
    'Opening a Workbook
    Dim workbook As IWorkbook = application.Workbooks.Open("Test.xls")
    Dim sheet As IWorksheet = workbook.Worksheets(0)
    'Creating Vba project
    Dim project As IVbaProject = workbook.VbaProject
    'Accessing vba modules collection
    Dim vbaModules As IVbaModules = project.Modules
    //Remove macro module
    vbaModules.Remove("Module1")
    'Saving as macro document
    workbook.SaveAs("Output.xlsm")
End Using
```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsm");
    openPicker.FileTypeFilter.Add(".xltm");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opening a workbook with a worksheet
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
    ExcelOpenType.Automatic);
    workbook.Version = ExcelVersion.Excel2016;
    IWorksheet worksheet = workbook.Worksheets[0];
    //Accessing Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Remove macro module
    vbaModules.Remove("Module1");
    // Save the Workbook
    StorageFile storageFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
    I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsm"
        });
        storageFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        storageFile = await local.CreateFileAsync("Output.xlsm",
        CreationCollisionOption.ReplaceExisting);
    }
    //Saving the workbook
    await workbook.SaveAsync(storageFile, ExcelSaveType.SaveAsMacro);
    // Launch the saved file
    await Windows.System.Launcher.LaunchFileAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Instantiate the excel application object.
    IApplication application = excelEngine.Excel;
    //Opening form module existing workbook

```

```

FileStream input = new FileStream(DataPathBase + "Test.xls", FileMode.Open,
    FileAccess.ReadWrite);
IWorkbook workbook = application.Workbooks.Open(input);
IWorksheet sheet = workbook.Worksheets[0];
//Accessing Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Remove macro module
vbaModules.Remove("Module1");
//Saving as macro document
FileStream output = new FileStream("Output.xlsm", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(output, ExcelSaveType.SaveAsMacro);
input.Close();
output.Close();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    // "App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream = assembly.GetManifestResourceStream("App.Test.xls");
    //Opening the workbook
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    workbook.Version = ExcelVersion.Excel2016;
    IWorksheet worksheet = workbook.Worksheets[0];
    //Accessing Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Remove macro module
    vbaModules.Remove("Module1");
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream, ExcelSaveType.SaveAsMacro);
    //Save the stream into XLSX file
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("sample.xlsm",
        "application/msexcel", stream);
}

```

RemoveAt(int index)

Vba module can be removed using the position from the IVbaModules collection.

- Index – Position of the Vba module in the modules collection.

The following code illustrate how to remove a macro using module index.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    // Instantiate the excel application object.

```

```

IApplication application = excelEngine.Excel;
// Opening a workbook
IWorkbook workbook = application.Workbooks.Open("Test.xls");
workbook.Version = ExcelVersion.Excel2016;
IWorksheet sheet = workbook.Worksheets[0];
//Accessing Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Remove macro module
vbaModules.RemoveAt(0);
//Saving as macro document
workbook.SaveAs("Output.xlsm");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
    'Instantiate the excel application object.
    Dim application As IApplication = excelEngine.Excel
    'Opening a Workbook
    Dim workbook As IWorkbook = application.Workbooks.Open("Test.xls")
    Dim sheet As IWorksheet = workbook.Worksheets(0)
    'Creating Vba project
    Dim project As IVbaProject = workbook.VbaProject
    'Accessing vba modules collection
    Dim vbaModules As IVbaModules = project.Modules
    //Remove macro module
    vbaModules.RemoveAt(0)
    'Saving as macro document
    workbook.SaveAs("Output.xlsm")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsm");
    openPicker.FileTypeFilter.Add(".xltn");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opening a workbook with a worksheet
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
        ExcelOpenType.Automatic);
    workbook.Version = ExcelVersion.Excel2016;
    IWorksheet worksheet = workbook.Worksheets[0];
    //Accessing Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Remove macro module
    vbaModules.RemoveAt(0);
}

```

```

// Save the Workbook
StorageFile storageFile;
if
(
  !(Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons"))
)
{
  FileSavePicker savePicker = new FileSavePicker();
  savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
  savePicker.SuggestedFileName = "Output";
  savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsm" });
  storageFile = await savePicker.PickSaveFileAsync();
}
else
{
  StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
  storageFile = await local.CreateFileAsync("Output.xlsm",
    CreationCollisionOption.ReplaceExisting);
}
//Saving the workbook
await workbook.SaveAsAsync(storageFile, ExcelSaveType.SaveAsMacro);
// Launch the saved file
await Windows.System.Launcher.LaunchFileAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
  //Instantiate the excel application object.
  IApplication application = excelEngine.Excel;
  //Opening form module existing workbook
  FileStream input = new FileStream(DataPathBase + "Test.xls", FileMode.Open,
    FileAccess.ReadWrite);
  IWorkbook workbook = application.Workbooks.Open(input);
  IWorksheet sheet = workbook.Worksheets[0];
  //Accessing Vba project
  IVbaProject project = workbook.VbaProject;
  //Accessing vba modules collection
  IVbaModules vbaModules = project.Modules;
  //Remove macro module
  vbaModules.RemoveAt(0);
  //Saving as macro document
  FileStream output = new FileStream("Output.xlsm", FileMode.Create,
    FileAccess.ReadWrite);
  workbook.SaveAs(output, ExcelSaveType.SaveAsMacro);
  input.Close();
  output.Close();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
  IApplication application = excelEngine.Excel;
  /"App" is the class of Portable project

```

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream("App.Test.xls");
//Opening the workbook
IWorkbook workbook = application.Workbooks.Open(inputStream);
workbook.Version = ExcelVersion.Excel2016;
IWorksheet worksheet = workbook.Worksheets[0];
//Accessing Vba project
IVbaProject project = workbook.VbaProject;
//Accessing vba modules collection
IVbaModules vbaModules = project.Modules;
//Remove macro module
vbaModules.RemoveAt(0);
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream, ExcelSaveType.SaveAsMacro);
//Save the stream into XLSX file
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("sample.xlsm",
"application/msexcel", stream);
}

```

Clear()

Clear() method removes all the Vba modules at once by clearing the module collection.

The following code illustrate how to remove all macros using clear method.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    // Instantiate the excel application object.
    IApplication application = excelEngine.Excel;
    // Opening a workbook
    IWorkbook workbook = application.Workbooks.Open("Test.xls");
    workbook.Version = ExcelVersion.Excel2016;
    IWorksheet sheet = workbook.Worksheets[0];
    //Accessing Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Remove all macros
    vbaModules.Clear();
    //Saving as macro document
    workbook.SaveAs("Output.xlsm");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
    'Instantiate the excel application object.
    Dim application As IApplication = excelEngine.Excel
    'Opening a Workbook
    Dim workbook As IWorkbook = application.Workbooks.Open("Test.xls")
    Dim sheet As IWorksheet = workbook.Worksheets(0)
    'Creating Vba project
    Dim project As IVbaProject = workbook.VbaProject
    'Accessing vba modules collection
    Dim vbaModules As IVbaModules = project.Modules

```

```
//Remove all macros
vbaModules.Clear()
'Saving as macro document
workbook.SaveAs ("Output.xlsm")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsm");
    openPicker.FileTypeFilter.Add(".xltn");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opening a workbook with a worksheet
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
        ExcelOpenType.Automatic);
    workbook.Version = ExcelVersion.Excel2016;
    IWorksheet worksheet = workbook.Worksheets[0];
    //Accessing Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Remove all macros
    vbaModules.Clear();
    // Save the Workbook
    StorageFile storageFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent ("Windows.Phone.U
        I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsm"
        });
        storageFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        storageFile = await local.CreateFileAsync("Output.xlsm",
            CreationCollisionOption.ReplaceExisting);
    }
    //Saving the workbook
    await workbook.SaveAsAsync(storageFile, ExcelSaveType.SaveAsMacro);
    // Launch the saved file
    await Windows.System.Launcher.LaunchFileAsync(storageFile);
}
```

ASP.NET CORE


```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    //Instantiate the excel application object.
    IApplication application = excelEngine.Excel;
    //Opening form module existing workbook
    FileStream input = new FileStream(DataPathBase + "Test.xls", FileMode.Open,
    FileAccess.ReadWrite);
    IWorkbook workbook = application.Workbooks.Open(input);
    IWorksheet sheet = workbook.Worksheets[0];
    //Accessing Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Remove all macros
    vbaModules.Clear();
    //Saving as macro document
    FileStream output = new FileStream("Output.xlsm", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(output, ExcelSaveType.SaveAsMacro);
    input.Close();
    output.Close();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream = assembly.GetManifestResourceStream("App.Test.xls");
    //Opening the workbook
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    workbook.Version = ExcelVersion.Excel2016;
    IWorksheet worksheet = workbook.Worksheets[0];
    //Accessing Vba project
    IVbaProject project = workbook.VbaProject;
    //Accessing vba modules collection
    IVbaModules vbaModules = project.Modules;
    //Remove all macros
    vbaModules.Clear();
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream, ExcelSaveType.SaveAsMacro);
    //Save the stream into XLSX file
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("sample.xlsm",
    "application/msexcel", stream);
}

```

SkipOnSave

SkipOnSave allows to resave the Excel document into normal XLSX and XLS documents.

The following code illustrate how to save the macro-enabled document into normal Excel document.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())

```

```
{
// Instantiate the excel application object.
IApplication application = excelEngine.Excel;
// Opening a workbook
IWorkbook workbook = application.Workbooks.Open("Test.xls");
workbook.Version = ExcelVersion.Excel2016;
IWorksheet sheet = workbook.Worksheets[0];
//Skip Macros while saving
application.SkipOnSave = SkipExtRecords.Macros;
//Saving as Excel without macro
workbook.SaveAs("Output.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
'Instantiate the excel application object.
Dim application As IApplication = excelEngine.Excel
'Opening a Workbook
Dim workbook As IWorkbook = application.Workbooks.Open("Test.xls")
Dim sheet As IWorksheet = workbook.Worksheets(0)
//Skip Macros while saving
application.SkipOnSave = SkipExtRecords.Macros
'Saving as Excel without macro
workbook.SaveAs("Output.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsm");
openPicker.FileTypeFilter.Add(".xltn");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opening a workbook with a worksheet
IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
workbook.Version = ExcelVersion.Excel2016;
IWorksheet worksheet = workbook.Worksheets[0];
//Skip Macros while saving
application.SkipOnSave = SkipExtRecords.Macros;
// Save the Workbook
StorageFile storageFile;
if
(! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons")))
{
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
```

```

savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
storageFile = await savePicker.PickSaveFileAsync();
}
else
{
StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
storageFile = await local.CreateFileAsync("Output.xlsx",
CreationCollisionOption.ReplaceExisting);
}
//Saving the workbook without macro
await workbook.SaveAsAsync(storageFile, ExcelSaveType.SaveAsXLS);
// Launch the saved file
await Windows.System.Launcher.LaunchFileAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
//Instantiate the excel application object.
IApplication application = excelEngine.Excel;
//Opening form module existing workbook
FileStream input = new FileStream(DataPathBase + "Test.xls", FileMode.Open,
FileAccess.ReadWrite);
IWorkbook workbook = application.Workbooks.Open(input);
IWorksheet sheet = workbook.Worksheets[0];
//Skip Macros while saving
application.SkipOnSave = SkipExtRecords.Macros;
//Saving as Excel without macro
FileStream output = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(output, ExcelSaveType.SaveAsXLS);
input.Close();
output.Close();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream("App.Test.xls");
//Opening the workbook
IWorkbook workbook = application.Workbooks.Open(inputStream);
workbook.Version = ExcelVersion.Excel2016;
IWorksheet worksheet = workbook.Worksheets[0];
//Skip Macros while saving
application.SkipOnSave = SkipExtRecords.Macros;
//Saving as Excel without macros
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream, ExcelSaveType.SaveAsXLS);
//Save the stream into XLSX file
}

```

```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("sample.xlsx",
"application/msexcel", stream);
}
```

Excel to PDF Conversion

XlsIO allows you to convert an entire workbook or a single worksheet into PDF document. Refer to the following links for assemblies/nuget packages required based on platforms to convert Excel document into PDF.

- [Assemblies Information](#)
- [NuGet Information](#)

Note: Excel to PDF conversion works proper in Blazor server-side alone and not in client-side.

Workbook to PDF

The following code illustrates how to convert an Excel workbook to PDF.

C#

```
using(ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
    ExcelOpenType.Automatic);
    //Open the Excel document to Convert
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Initialize PDF document
    PdfDocument pdfDocument = new PdfDocument();
    //Convert Excel document into PDF document
    pdfDocument = converter.Convert();
    //Save the PDF file
    pdfDocument.Save("ExcelToPDF.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
'Open the Excel document to convert
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Initialize the PDF document
Dim pdfDocument As PdfDocument = New PdfDocument()
'Convert Excel document into PDF document
pdfDocument = converter.Convert()
'Save the PDF file
pdfDocument.Save("ExcelToPDF.pdf")
End Using
```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
#region Excel To PDF
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream = assembly.GetManifestResourceStream("ExcelToPDF.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(excelStream);
    //Initialize XlsIO renderer.
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert Excel document into PDF document
    PdfDocument pdfDocument = renderer.ConvertToPDF(workbook);
    //Save the PDF document to stream.
    MemoryStream stream = new MemoryStream();
    await doc.SaveAsync(stream);
    Save(stream, "ExcelToPDF.pdf");
    excelStream.Dispose();
    stream.Dispose();
}
#endregion
//Save the workbook stream as a file.
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Sample";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
            ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
        stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
}

```

```
#endregion
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("ExcelToPDF.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIO renderer.
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert Excel document into PDF document
    PdfDocument pdfDocument = renderer.ConvertToPDF(workbook);
    Stream stream = new FileStream("ExcelToPDF.pdf", FileMode.Create,
    FileAccess.ReadWrite);
    pdfDocument.Save(stream);
    excelStream.Dispose();
    stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream = assembly.GetManifestResourceStream("ExcelToPDF.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIO renderer.
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert Excel document into PDF document
    PdfDocument pdfDocument = renderer.ConvertToPDF(workbook);
    //Save the PDF document to stream.
    MemoryStream stream = new MemoryStream();
    doc.Save(stream);
    stream.Position = 0;
    //Save the stream into pdf file
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("ExcelToPDF.pdf", "application/pdf", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().Save("ExcelToPDF.pdf", "application/pdf", stream);
    }
    excelStream.Dispose();
    stream.Dispose();
}
```

To learn more about different conversion settings in Excel To PDF conversion, refer to the ExcelToPdfConverterSettings in API section.

Worksheet to PDF

The following code shows how to convert a particular sheet to PDF document.

C#

```
Using(ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //convert the sheet to PDF
    ExcelToPdfConverter converter = new ExcelToPdfConverter(sheet);
    PdfDocument pdfDocument= new PdfDocument();
    pdfDocument = converter.Convert();
    pdfDocument.Save("ExcelToPDF.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
    ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Converts the particular sheet
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(sheet)
Dim pdfDocument As PdfDocument = New PdfDocument()
pdfDocument = converter.Convert()
'Save the PDF file
pdfDocument.Save("ExcelToPDF.pdf")
End Using
```

UWP

```
//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
#region Excel To PDF
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream = assembly.GetManifestResourceStream("ExcelToPDF.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(excelStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Initialize XlsIO renderer.
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert Excel document into PDF document
    PdfDocument pdfDocument = renderer.ConvertToPDF(worksheet);
}
```

```

//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
Save(stream, "ExcelToPDF.pdf");
excelStream.Dispose();
stream.Dispose();
}
#endregion
//Save the workbook stream as a file.
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Sample";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
            ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
        stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
}
#endregion

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("ExcelToPDF.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Initialize XlsIO renderer.
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert Excel document into PDF document
}

```



```
PdfDocument pdfDocument = renderer.ConvertToPDF(worksheet);
Stream stream = new FileStream("ExcelToPDF.pdf", FileMode.Create,
    FileAccess.ReadWrite);
pdfDocument.Save(stream);
excelStream.Dispose();
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream = assembly.GetManifestResourceStream("ExcelToPDF.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Initialize XlsIO renderer.
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert Excel document into PDF document
    PdfDocument pdfDocument = renderer.ConvertToPDF(worksheet);
    //Save the PDF document to stream.
    MemoryStream stream = new MemoryStream();
    doc.Save(stream);
    stream.Position = 0;
    //Save the stream into pdf file
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("ExcelToPDF.pdf", "application/pdf", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().Save("ExcelToPDF.pdf", "application/pdf", stream);
    }
    excelStream.Dispose();
    stream.Dispose();
}
```

Creating individual PDF document for each worksheet

The following code snippet shows how to create an individual PDF document for each worksheet in a workbook.

C#

```
Using(ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
```

```

PdfDocument pdfDocument = new PdfDocument();
foreach (IWorksheet sheet in workbook.Worksheets)
{
    ExcelToPdfConverter converter = new ExcelToPdfConverter(sheet);
    pdfDocument = converter.Convert();
    //Save the PDF file
    pdfDocument.Save(sheet.Name+".pdf");
    converter.Dispose();
}
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim pdfDocument As New PdfDocument()
For Each sheet As IWorksheet In workbook.Worksheets
    Dim converter As New ExcelToPdfConverter(sheet)
    PdfDocument = converter.Convert()
    'Save the PDF file
    PdfDocument.Save(sheet.Name + ".pdf")
    converter.Dispose()
Next
End Using

```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard assemblies in UWP platform
#region Excel To PDF
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream = assembly.GetManifestResourceStream("ExcelToPDF.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(excelStream);
    //Initialize XlsIO renderer.
    XlsIORenderer renderer = new XlsIORenderer();
    PdfDocument pdfDocument = new PdfDocument();
    foreach (IWorksheet sheet in workbook.Worksheets)
    {
        pdfDocument = renderer.ConvertToPDF(sheet);
        //Save the PDF file
        MemoryStream stream = new MemoryStream();
        await doc.SaveAsync(stream);
        Save(stream, sheet.Name+".pdf");
        stream.Dispose();
    }
    excelStream.Dispose();
}
#endregion
//Save the workbook stream as a file.

```

```

#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Sample";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
            ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
        stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
}
#endregion

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("ExcelToPDF.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIO renderer.
    XlsIORenderer renderer = new XlsIORenderer();
    PdfDocument pdfDocument = new PdfDocument();
    foreach (IWorksheet sheet in workbook.Worksheets)
    {
        pdfDocument = renderer.ConvertToPDF(sheet);
        //Save the PDF file
        Stream stream = new FileStream(sheet.Name+".pdf", FileMode.Create,
            FileAccess.ReadWrite);
        pdfDocument.Save(stream);
        stream.Dispose();
    }
    excelStream.Dispose();
}

```

```
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream = assembly.GetManifestResourceStream("ExcelToPDF.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIO renderer.
    XlsIORenderer renderer = new XlsIORenderer();
    PdfDocument pdfDocument = new PdfDocument();
    foreach (IWorksheet sheet in workbook.Worksheets)
    {
        pdfDocument = renderer.ConvertToPDF(sheet);
        //Save the PDF file
        Stream stream = new FileStream(sheet.Name+".pdf", FileMode.Create,
            FileAccess.ReadWrite);
        pdfDocument.Save(stream);
        stream.Position = 0;
        //Save the stream into pdf file
        if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
            TargetPlatform.Windows)
        {
            Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("ExcelToPDF.pdf", "application/pdf", stream);
        }
        else
        {
            Xamarin.Forms.DependencyService.Get<ISave>().Save("ExcelToPDF.pdf", "application/pdf", stream);
        }
        stream.Dispose();
    }
    excelStream.Dispose();
}
```

Excel with chart to PDF

XlsIO allows you to convert a workbook/worksheet with charts or a single chart into PDF document.

To preserve the charts during Excel To PDF conversion in .NET Framework, initialize the **ChartToImageConverter** of **IApplication** interface otherwise the charts present in worksheet gets skipped.

The following code illustrates how to convert an Excel with chart to PDF document.

C#

```
Using(ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
```

```

//Instantiating the ChartToImageConverter and assigning the
ChartToImageConverter instance of XlsIO application
application.ChartToImageConverter = new ChartToImageConverter();
//Tuning chart image quality
application.ChartToImageConverter.ScalingMode = ScalingMode.Best;
IWorkbook workbook = application.Workbooks.Open("chart.xlsx");
IWorksheet worksheet = workbook.Worksheets[0];
ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
PdfDocument pdfDocument = new PdfDocument();
pdfDocument = converter.Convert();
pdfDocument.Save("ExcelToPDF.pdf");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
'Instantiating the ChartToImageConverter and assigning the
ChartToImageConverter instance of XlsIO application
application.ChartToImageConverter = New ChartToImageConverter()
'Tuning chart image quality
application.ChartToImageConverter.ScalingMode = ScalingMode.Best
Dim workbook As IWorkbook = application.Workbooks.Open("chart.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim converter As New ExcelToPdfConverter(workbook)
Dim pdfDocument As New PdfDocument()
pdfDocument = converter.Convert()
pdfDocument.Save("ExcelToPDF.pdf")
End Using

```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard 2.0
assemblies in UWP platform
#region Excel To PDF
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Initializing XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from an embedded resource collection
Stream excelStream = assembly.GetManifestResourceStream("chart.xlsx");
IWorkbook workbook = await application.Workbooks.OpenAsync(excelStream);
//Convert Excel document with charts into PDF document
PdfDocument pdfDocument = renderer.ConvertToPDF(workbook);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await doc.SaveAsync(stream);
Save(stream, "ExcelToPDF.pdf");
excelStream.Dispose();
stream.Dispose();
}
#endregion

```

```

//Save the workbook stream as a file.
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Sample";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
            ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
            stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
}
#endregion

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Initialize XlsIO renderer.
    XlsIORenderer renderer = new XlsIORenderer();
    FileStream excelStream = new FileStream("chart.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Convert Excel document with charts into PDF document
    PdfDocument pdfDocument = renderer.ConvertToPDF(workbook);
    Stream stream = new FileStream("ExcelToPDF.pdf", FileMode.Create,
        FileAccess.ReadWrite);
    pdfDocument.Save(stream);
    excelStream.Dispose();
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Initialize XlsIO renderer.
    XlsIORenderer renderer = new XlsIORenderer();
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream = assembly.GetManifestResourceStream("chart.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Convert Excel document into PDF document
    PdfDocument pdfDocument = renderer.ConvertToPDF(workbook);
    //Save the PDF document to stream.
    MemoryStream stream = new MemoryStream();
    doc.Save(stream);
    stream.Position = 0;
    //Save the stream into pdf file
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("ExcelToPDF.pdf", "application/pdf", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().Save("ExcelToPDF.pdf", "application/pdf", stream);
    }
    excelStream.Dispose();
    stream.Dispose();
}

```

Substitute Font in Excel-to-PDF Conversion

By default, XlsIO substitutes unsupported fonts to Microsoft Sans Serif in Excel-to-PDF conversion. However, you may require substituting a different font or the same font for the unsupported font during the conversion. XlsIO supports substituting unsupported or missing fonts from the event

SubstituteFont. The event has the below arguments:

AlternateFontName – Substitutes an available font in the machine for the **OriginalFontName**.

AlternateFontStream – Substitutes a font from stream that is added as embedded resource for the **OriginalFontName**.

The following code illustrates how to perform Excel-to-PDF conversion by substituting unsupported fonts in the machine.

C#

```

using Syncfusion.ExcelToPdfConverter;
using Syncfusion.Pdf;
using Syncfusion.XlsIO;
using Syncfusion.XlsIO.Implementation;
using System.IO;
using System.Reflection;

```

```

namespace FontSubstitution
{
    class Program
    {
        static void Main(string[] args)
        {
            using (ExcelEngine excelEngine = new ExcelEngine())
            {
                IApplication application = excelEngine.Excel;
                application.DefaultVersion = ExcelVersion.Xlsx;
                //Initializes the SubstituteFont event to perform font substitution in
                Excel-to-PDF conversion.
                application.SubstituteFont += new
                SubstituteFontEventHandler(SubstituteFont);
                IWorkbook workbook = application.Workbooks.Open("Template.xlsx");
                IWorksheet worksheet = workbook.Worksheets[0];
                ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
                PdfDocument pdfDocument = new PdfDocument();
                pdfDocument = converter.Convert();
                pdfDocument.Save("ExcelToPDF.pdf");
            }
        }

        private static void SubstituteFont(object sender, SubstituteFontEventArgs
        args)
        {
            //Substitute a font if the specified font is not installed in the machine.
            if (args.OriginalFontName == "Arial Unicode MS")
            {
                //Substitute by font name.
                args.AlternateFontName = "Arial";
            }
            else if (args.OriginalFontName == "Homizio")
            {
                //Substitute by font stream.
                var assembly = Assembly.GetExecutingAssembly();
                var resourceName = "ExceltoPDF.Fonts.Homizio.ttf";
                Stream fileStream = assembly.GetManifestResourceStream(resourceName);
                MemoryStream memoryStream = new MemoryStream();
                fileStream.CopyTo(memoryStream);
                fileStream.Close();
                args.AlternateFontStream = memoryStream;
            }
        }
    }
}

```

VB.NET

```

Imports Syncfusion.ExcelToPdfConverter
Imports Syncfusion.Pdf
Imports Syncfusion.XlsIO
Imports Syncfusion.XlsIO.Implementation
Imports System.IO
Imports System.Reflection
Namespace FontSubstitution
Class Program

```



```

Private Shared Sub Main(ByVal args() As String)
Dim excelEngine As ExcelEngine = New ExcelEngine
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Xlsx
'Initializes the SubstituteFont event to perform font substitution in Excel-
to-PDF conversion.
AddHandler application.SubstituteFont, AddressOf Me.SubstituteFont
Dim workbook As IWorkbook = application.Workbooks.Open("Template.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
Dim pdfDocument As PdfDocument = New PdfDocument
pdfDocument = converter.Convert
pdfDocument.Save("ExcelToPDF.pdf")
End Sub
Private Shared Sub SubstituteFont(ByVal sender As Object, ByVal args As
SubstituteFontEventArgs)
'Substitute a font if the specified font is not installed in the machine.
If (args.OriginalFontName = "Arial Unicode MS") Then
'Substitute by font name.
args.AlternateFontName = "Arial"
ElseIf (args.OriginalFontName = "Homizio") Then
'Substitute by font stream.
Dim assembly = Assembly.GetExecutingAssembly
Dim resourceName = "ExceltoPDF.Fonts.Homizio.ttf"
Dim fileStream As Stream = assembly.GetManifestResourceStream(resourceName)
Dim memoryStream As MemoryStream = New MemoryStream
fileStream.CopyTo(memoryStream)
fileStream.Close
args.AlternateFontStream = memoryStream
End If
End Sub
End Class
End Namespace

```

UWP

```

//Excel-to-PDF conversion can be performed by referring .NET Standard 2.0
assemblies in UWP platform
using System;
using System.Collections.Generic;
using System.IO;
using System.Reflection;
using Windows.Storage;
using Windows.Storage.Pickers;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Syncfusion.XlsIORenderer;
using Syncfusion.Pdf;
using Syncfusion.XlsIO;
using Syncfusion.XlsIO.Implementation;
namespace FontSubstitution
{
public sealed partial class MainPage : Page
{
public MainPage()
{

```

```

this.InitializeComponent();
}
private async void button_Click(object sender, RoutedEventArgs e)
{
    #region Excel To PDF
    using (ExcelEngine excelEngine = new ExcelEngine())
    {
        IApplication application = excelEngine.Excel;
        //Initializes the SubstituteFont event to perform font substitution during
        //Excel-to-PDF conversion
        application.SubstituteFont += new
        SubstituteFontEventHandler(SubstituteFont);
        //Initializing XlsIORenderer
        XlsIORenderer renderer = new XlsIORenderer();
        //Gets assembly
        Assembly assembly = typeof(App).GetTypeInfo().Assembly;
        //Gets input Excel document from an embedded resource collection
        Stream excelStream = assembly.GetManifestResourceStream("Template.xlsx");
        IWorkbook workbook = await application.Workbooks.OpenAsync(excelStream);
        //Convert Excel document with charts into PDF document
        PdfDocument pdfDocument = renderer.ConvertToPDF(workbook);
        //Save the PDF document to stream.
        MemoryStream stream = new MemoryStream();
        await pdfDocument.SaveAsync(stream);
        Save(stream, "ExcelToPDF.pdf");
        excelStream.Dispose();
        stream.Dispose();
    }
    #endregion
}
private static void SubstituteFont(object sender, SubstituteFontEventArgs
args)
{
    //Substitute a font if the specified font is not installed in the machine.
    if (args.OriginalFontName == "Arial Unicode MS")
    {
        args.AlternateFontName = "Arial";
    }
    else if (args.OriginalFontName == "Homizior")
    {
        //Substitute by font stream.
        Assembly assembly = typeof(App).GetTypeInfo().Assembly;
        Stream fileStream = assembly.GetManifestResourceStream("Homizior.ttf");
        MemoryStream memoryStream = new MemoryStream();
        fileStream.CopyTo(memoryStream);
        fileStream.Close();
        args.AlternateFontStream = memoryStream;
    }
}
//Save the workbook stream as a file.
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;

```

```

if
(
    !(Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons"))
)
{
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.DefaultFileExtension = ".pdf";
    savePicker.SuggestedFileName = "Sample";
    savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() { ".pdf" });
    stFile = await savePicker.PickSaveFileAsync();
}
else
{
    StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
    stFile = await local.CreateFileAsync(filename,
        CreationCollisionOption.ReplaceExisting);
}
if (stFile != null)
{
    Windows.Storage.Streams.IRandomAccessStream fileStream = await
    stFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream st = fileStream.AsStreamForWrite();
    st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
    st.Flush();
    st.Dispose();
    fileStream.Dispose();
}
}
#endregion
}
}

```

ASP.NET CORE

```

using System;
using System.IO;
using Syncfusion.XlsIO;
using Syncfusion.XlsIORenderer;
using Syncfusion.Pdf;
using System.Reflection;
namespace FontSubstitution
{
    class Program
    {
        static void Main(string[] args)
        {
            using (ExcelEngine excelEngine = new ExcelEngine())
            {
                IApplication application = excelEngine.Excel;
                //Initializes the SubstituteFont event to perform font substitution during
                Excel-to-PDF conversion
                application.SubstituteFont += new
                SubstituteFontEventHandler(SubstituteFont);
                //Initialize XlsIO renderer.
                XlsIORenderer renderer = new XlsIORenderer();
            }
        }
    }
}

```

```

FileStream excelStream = new FileStream("Template.xlsx", FileMode.Open,
    FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(excelStream);
//Convert Excel document with charts into PDF document
PdfDocument pdfDocument = renderer.ConvertToPDF(workbook);
Stream stream = new FileStream("ExcelToPDF.pdf", FileMode.Create,
    FileAccess.ReadWrite);
pdfDocument.Save(stream);
excelStream.Dispose();
stream.Dispose();
}
}
private static void SubstituteFont(object sender, SubstituteFontEventArgs
args)
{
    //Substitute a font if the specified font is not installed in the machine.
    if (args.OriginalFontName == "Arial Unicode MS")
        args.AlternateFontName = "Arial";
    else if (args.OriginalFontName == "Homizio")
    {
        //Substitute by font stream.
        var assembly = Assembly.GetExecutingAssembly();
        var resourceName = "ExceltoPDF.Fonts.Homizio.ttf";
        Stream fileStream = assembly.GetManifestResourceStream(resourceName);
        MemoryStream memoryStream = new MemoryStream();
        fileStream.CopyTo(memoryStream);
        fileStream.Close();
        args.AlternateFontStream = memoryStream;
    }
}
}
}
}

```

XAMARIN

```

using System;
using Xamarin.Forms;
using System.IO;
using Syncfusion.XlsIO;
using Syncfusion.XlsIORenderer;
using Syncfusion.Pdf;
using Syncfusion.XlsIO.Implementation;
using System.Reflection;
namespace FontSubstitution
{
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            InitializeComponent();
        }
        void OnButtonClicked(object sender, EventArgs args)
        {
            using (ExcelEngine excelEngine = new ExcelEngine())
            {
                IApplication application = excelEngine.Excel;
            }
        }
    }
}

```

```

//Initializes the SubstituteFont event to perform font substitution during
Excel-to-PDF conversion
application.SubstituteFont += new
SubstituteFontEventHandler(SubstituteFont);
//Initialize XlsIO renderer.
XlsIORenderer renderer = new XlsIORenderer();
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from an embedded resource collection
Stream excelStream = assembly.GetManifestResourceStream("Template.xlsx");
IWorkbook workbook = application.Workbooks.Open(excelStream);
//Convert Excel document into PDF document
PdfDocument pdfDocument = renderer.ConvertToPDF(workbook);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
pdfDocument.Save(stream);
stream.Position = 0;
//Save the stream into pdf file
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("ExcelToPDF.pdf",
"application/pdf", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().Save("ExcelToPDF.pdf",
"application/pdf", stream);
}
excelStream.Dispose();
stream.Dispose();
}
}
private static void SubstituteFont(object sender, SubstituteFontEventArgs
args)
{
//Substitute a font if the specified font is not installed in the machine.
if (args.OriginalFontName == "Arial Unicode MS")
args.AlternateFontName = "Arial";
else if (args.OriginalFontName == "Homizio")
{
//Substitute by font stream.
Stream fileStream =
typeof(App).GetTypeInfo().Assembly.GetManifestResourceStream("ExceltoPDF.Fon
ts.Homizio.ttf");
MemoryStream memoryStream = new MemoryStream();
fileStream.CopyTo(memoryStream);
fileStream.Close();
args.AlternateFontStream = memoryStream;
}
}
}
}

```

Excel to PDF conversion in Linux OS

In Linux OS, you can perform the Excel to PDF conversion using .NET Core (Targeting .netcoreapp) application. You can refer [Excel to PDF conversion NuGet packages](#) to know about the packages required to deploy .NET Core (Targeting .netcoreapp) application with Excel to PDF conversion capabilities.

In addition to the previous NuGet packages, SkiaSharp.Linux helper NuGet package is required that can be generated by the following steps:

1. Download libSkiaSharp.so [here](#).
2. Create a folder and name it as SkiaSharp.Linux and place the downloaded file in the folder structure "SkiaSharp.Linux\runtimes\linux-x64\native"
3. Create a nuspec file with name SkiaSharp.Linux.nuspec using the following metadata information and place it inside SkiaSharp.Linux folder. The nuspec file can be customized.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<package xmlns="http://schemas.microsoft.com/packaging/2012/06/nuspec.xsd">
  <metadata>
    <id>SkiaSharp.Linux</id>
    <version>1.59.3</version>
    <title>SkiaSharp for Linux</title>
    <authors>Syncfusion Inc.</authors>
    <owners>Syncfusion Inc.</owners>
    <requireLicenseAcceptance>false</requireLicenseAcceptance>
    <description>SkiaSharp for Linux is a supporting package for Linux
platforms.</description>
    <tags>linux, cross-platform, skiasharp, net-standard, net-core, excel-to-
pdf</tags>
    <dependencies>
      <group targetFramework=".NETStandard1.4">
        <dependency id="SkiaSharp" version="1.59.3" />
      </group>
    </dependencies>
  </metadata>
</package>
```

4. Make sure that the nuget.exe file is present along with SkiaSharp.Linux folder (in the parent folder of SkiaSharp.Linux folder). If not, download it from [here](#).
5. Open a command prompt and navigate to SkiaSharp.Linux folder.
6. Execute the following command.

C#

```
Using(ExcelEngine excelEngine = new ExcelEngine())
{
  IApplication application = excelEngine.Excel;
  IWorkbook workbook = application.Workbooks.Open("Excel.xlsx");
  // Convert the workbook
  ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
  // Print the converted PDF document
  converter.Print();
}
```

```
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Excel.xlsx")
'Convert the workbook
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Print the converted PDF document
converter.Print()
End Using
```

UWP

```
//XlsIO supports Excel To PDF conversion in Windows Forms, WPF, ASP.NET, and
ASP.NET MVC platforms. Refer to the Workbook to PDF section to convert using
web service.
```

ASP.NET CORE

```
//XlsIO supports Excel To PDF conversion in Windows Forms, WPF, ASP.NET, and
ASP.NET MVC platforms. Refer to the Workbook to PDF section to convert using
web service.
```

XAMARIN

```
//XlsIO supports Excel To PDF conversion in Windows Forms, WPF, ASP.NET, and
ASP.NET MVC platforms. Refer to the Workbook to PDF section to convert using
web service.
```

Print with printer settings

The following code snippet illustrates how to print the Excel document with printer settings in XlsIO.

C#

```
Using(excelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("Excel.xlsx");
//Convert the workbook
ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
//Initialize the printer settings
PrinterSettings printerSettings = new PrinterSettings();
//customizing the printer settings
printerSettings.PrinterName = "HP LaserJet Pro MFP M127-M128 PCLmS";
printerSettings.Copies = 2;
printerSettings.FromPage = 2;
printerSettings.ToPage = 3;
printerSettings.DefaultPageSettings.Color = false;
printerSettings.Duplex = Duplex.Vertical;
//Print the converted PDF document with printer settings
converter.Print(printerSettings);
}
```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Excel.xlsx")
'Convert the workbook
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Initialize the printer settings
Dim printerSettings As PrinterSettings = New PrinterSettings
'customizing the printer settings
printerSettings.PrinterName = "HP LaserJet Pro MFP M127-M128 PCLmS"
printerSettings.Copies = 2
printerSettings.FromPage = 2
printerSettings.ToPage = 3
printerSettings.DefaultPageSettings.Color = false
printerSettings.Duplex = Duplex.Vertical
'Print the converted PDF document with printer settings
converter.Print(printerSettings)
End Using

```

UWP

```

//XlsIO supports Excel To PDF conversion in Windows Forms, WPF, ASP.NET, and  
ASP.NET MVC platforms. Refer to the Workbook to PDF section to convert using  
web service.

```

ASP.NET CORE

```

//XlsIO supports Excel To PDF conversion in Windows Forms, WPF, ASP.NET, and  
ASP.NET MVC platforms. Refer to the Workbook to PDF section to convert using  
web service.

```

XAMARIN

```

//XlsIO supports Excel To PDF conversion in Windows Forms, WPF, ASP.NET, and  
ASP.NET MVC platforms. Refer to the Workbook to PDF section to convert using  
web service.

```

Print with Excel To PDF converter settings

The following code snippet illustrates how to print the Excel document with Excel To PDF converter settings in XlsIO.

C#

```

Using(ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Excel.xlsx");
    //Convert the workbook
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Initializes the Excel To PDF converter setting class
    ExcelToPdfConverterSettings converterSettings = new
    ExcelToPdfConverterSettings();
}

```



```
//Layout the page using FitAllColumnsOnOnePage options
converterSettings.DisplayGridLines = GridLinesDisplayStyle.Visible;
converterSettings.LayoutOptions = LayoutOptions.FitAllColumnsOnOnePage;
//Print the converted PDF document with converter settings
converter.Print(converterSettings);
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Excel.xlsx")
'Convert the workbook
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Initialize the Excel To PDF converter setting class
Dim converterSettings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings
'Layout the page using FitAllColumnsOnOnePage options
converterSettings.DisplayGridLines = GridLinesDisplayStyle.Visible
converterSettings.LayoutOptions = LayoutOptions.FitAllColumnsOnOnePage
'Print the converted PDF document with converter settings
converter.Print(converterSettings)
End Using
```

UWP

```
//XlsIO supports Excel To PDF conversion in Windows Forms, WPF, ASP.NET, and
ASP.NET MVC platforms. Refer to the Workbook to PDF section to convert using
web service.
```

ASP.NET CORE

```
//XlsIO supports Excel To PDF conversion in Windows Forms, WPF, ASP.NET, and
ASP.NET MVC platforms. Refer to the Workbook to PDF section to convert using
web service.
```

XAMARIN

```
//XlsIO supports Excel To PDF conversion in Windows Forms, WPF, ASP.NET, and
ASP.NET MVC platforms. Refer to the Workbook to PDF section to convert using
web service.
```

Print with Excel To PDF converter and printer settings

The following code snippet illustrates how to print the Excel document with Excel To PDF converter settings and printer settings in XlsIO.

C#

```
Using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
IWorkbook workbook = application.Workbooks.Open("Excel.xlsx");
ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
//Initialize the printer settings
```

```

PrinterSettings printerSettings = new PrinterSettings();
//customizing the printer settings
printerSettings.PrinterName = "HP LaserJet Pro MFP M127-M128 PCLmS";
printerSettings.Copies = 2;
printerSettings.FromPage = 2;
printerSettings.ToPage = 3;
printerSettings.DefaultPageSettings.Color = true;
printerSettings.Duplex = Duplex.Vertical;
printerSettings.Collate = true;
//Initializes the Excel To PDF converter setting class
ExcelToPdfConverterSettings converterSettings = new
ExcelToPdfConverterSettings();
//Layout the page using FitAllColumnsOnOnePage options
converterSettings.DisplayGridLines = GridLinesDisplayStyle.Visible;
converterSettings.LayoutOptions = LayoutOptions.FitAllColumnsOnOnePage;
//Print the converted PDF document with printer settings and converter
settings
converter.Print(printerSettings, converterSettings);
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Excel.xlsx")
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Initialize the printer settings
Dim printerSettings As PrinterSettings = New PrinterSettings
'customizing the printer settings
printerSettings.PrinterName = "HP LaserJet Pro MFP M127-M128 PCLmS"
printerSettings.Copies = 2
printerSettings.FromPage = 2
printerSettings.ToPage = 3
printerSettings.DefaultPageSettings.Color = true
printerSettings.Duplex = Duplex.Vertical
printerSettings.Collate = true
'Initialize the Excel To PDF converter setting class
Dim converterSettings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings
'Layout the page using FitAllColumnsOnOnePage options
converterSettings.DisplayGridLines = GridLinesDisplayStyle.Visible
converterSettings.LayoutOptions = LayoutOptions.FitAllColumnsOnOnePage
'Print the converted PDF document with printer settings and converter
settings
converter.Print(printerSettings, converterSettings)
End Using

```

UWP

```

//XlsIO supports Excel To PDF conversion in Windows Forms, WPF, ASP.NET, and
ASP.NET MVC platforms. Refer to the Workbook to PDF section to convert using
web service.

```

ASP.NET CORE

```
//XlsIO supports Excel To PDF conversion in Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms. Refer to the Workbook to PDF section to convert using web service.
```

XAMARIN

```
//XlsIO supports Excel To PDF conversion in Windows Forms, WPF, ASP.NET, and ASP.NET MVC platforms. Refer to the Workbook to PDF section to convert using web service.
```

Note: Printing support is applicable only in Windows Forms and WPF platforms.

Supported elements

This feature supports the following elements:

- Styles
- Rich-text formatting
- Headers and footers
- Images
- Picture recolor
- Black and white
- Color change
- DuoTone
- Gray scale
- Text boxes
- Hyperlinks
- Document properties
- Table styles
- Text rotations
- Excel page setup options
- Unicode
- Print titles
- Page breaks
- Print area
- Print order
- 2D charts
- 3D charts
- AutoShapes
- Group shapes
- Conditional formats
- Pivot tables

Unsupported elements

The following list contains unsupported elements that presently not preserved in the generated PDF document:

- Gradient fill
- Comments

- Sparklines
- Pivot charts
- SmartArt graphics
- Different first page headers
- Different odd and even pages
- Conditional formats
- Data bars
- Color scales
- Tables
- Custom styles
- Row and column headings
- Form controls
- ActiveX controls
- OLE objects

Excel to PDF Converter Settings

XlsIO allows you to convert an entire workbook or a single worksheet into PDF document with conversion settings of Excel to PDF converter.

Auto-detect Complex Script

This property enables the complex script validation for the text present in Excel document and renders it into PDF document. Its default value is FALSE.

The following complete code snippet explains how to enable [AutoDetectComplexScript](#) property while converting an Excel document to PDF.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Enable AutoDetectComplexScript property
    settings.AutoDetectComplexScript = true;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
```

```

Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Enable AutoDetectComplexScript property
settings.AutoDetectComplexScript = True
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using

```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Get input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Enable AutoDetectComplexScript property
    settings.AutoDetectComplexScript = true;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    Save(stream, "Output.pdf");
    excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
}

```

```

}
else
{
    StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
    stFile = await local.CreateFileAsync(filename,
        CreationCollisionOption.ReplaceExisting);
}
if (stFile != null)
{
    Windows.Storage.Streams.IRandomAccessStream fileStream = await
    stFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream st = fileStream.AsStreamForWrite();
    st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
    st.Flush();
    st.Dispose();
    fileStream.Dispose();
}
stream.Dispose();
}
#endregion

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Enable AutoDetectComplexScript property
    settings.AutoDetectComplexScript = true;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Download the PDF file in the browser
    FileStreamResult fileStreamResult = new FileStreamResult(stream,
        "application/pdf");
    fileStreamResult.FileNameDownload = "Output.pdf";
    return fileStreamResult;
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
}

```

```
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Enable AutoDetectComplexScript property
settings.AutoDetectComplexScript = true;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}
```

{% endtabs%}

Custom Paper Size

This property helps to set the required paper size in inches. The default value is empty(i.e.,{Width = 0.0 Height = 0.0}).

The following complete code snippet explains how to set [CustomPaperSize](#) while converting Excel document to PDF.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Set CustomerPaperSize
    settings.CustomPaperSize = new SizeF(10, 20);
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Set CustomerPaperSize
settings.CustomPaperSize = New SizeF(10, 20)
'Load the Excel document into ExcelToPdfConverter
```

```

Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using

```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Get input Excel document from an embedded resource collection
    Stream excelStream =
        assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set CustomerPaperSize
    settings.CustomPaperSize = new SizeF(10, 20);
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    Save(stream, "Output.pdf");
    excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (
        !(Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons"))
    )
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() { ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
    }
}

```



```

stFile = await local.CreateFileAsync(filename,
CreationCollisionOption.ReplaceExisting);
}
if (stFile != null)
{
Windows.Storage.Streams.IRandomAccessStream fileStream = await
stFile.OpenAsync(FileAccessMode.ReadWrite);
Stream st = fileStream.AsStreamForWrite();
st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
st.Flush();
st.Dispose();
fileStream.Dispose();
}
stream.Dispose();
}
#endregion

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(excelStream);
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Set CustomerPaperSize
settings.CustomPaperSize = new SizeF(10, 20);
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Saving the Excel to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF file in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from an embedded resource collection
Stream excelStream =
assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
}

```

```

IWorkbook workbook = application.Workbooks.Open(excelStream);
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Set CustomerPaperSize
settings.CustomPaperSize = new SizeF(10, 20);
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}

```

Display Gridlines

This property helps to set the gridlines display style in Excel to PDF converted document. Three display styles are **Auto**, **Invisible** and **Visible**. These are maintained under [GridLinesDisplayStyle](#) enumeration.

Auto

Gridlines will not be rendered in the output PDF document, when display style is selected as **Auto**. Its default value is Invisible. The following code snippet explains how to select **Auto** display style for gridlines.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Set the gridlines display style as Auto
    settings.DisplayGridLines = GridLinesDisplayStyle.Auto;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Set the gridlines display style as Auto

```

```

settings.DisplayGridLines = GridLinesDisplayStyle.Auto
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using

```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Get input Excel document from an embedded resource collection
    Stream excelStream =
        assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set the gridlines display style as Auto
    settings.DisplayGridLines = GridLinesDisplayStyle.Auto;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    Save(stream, "Output.pdf");
    excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
            ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {

```

```

StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
stFile = await local.CreateFileAsync(filename,
CreationCollisionOption.ReplaceExisting);
}
if (stFile != null)
{
Windows.Storage.Streams.IRandomAccessStream fileStream = await
stFile.OpenAsync(FileAccessMode.ReadWrite);
Stream st = fileStream.AsStreamForWrite();
st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
st.Flush();
st.Dispose();
fileStream.Dispose();
}
stream.Dispose();
}
#endregion

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(excelStream);
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Set the gridlines display style as Auto
settings.DisplayGridLines = GridLinesDisplayStyle.Auto;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Saving the Excel to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF file in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from an embedded resource collection

```

```

Stream excelStream =
assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(excelStream);
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Set the gridlines display style as Auto
settings.DisplayGridLines = GridLinesDisplayStyle.Auto;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}

```

Invisible

Gridlines will not be rendered in the output PDF document, when display style is selected as **Invisible**. The following code snippet explains how to select **Invisible** display style for gridlines.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
//Initialize ExcelToPdfConverterSettings
ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
//Set the gridlines display style as Invisible
settings.DisplayGridLines = GridLinesDisplayStyle.Invisible;
//Load the Excel document into ExcelToPdfConverter
ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
//Convert the Excel document to PDF with converter settings
PdfDocument document = converter.Convert(settings);
//Save the PDF document
document.Save("Output.pdf");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Set the gridlines display style as Invisible
settings.DisplayGridLines = GridLinesDisplayStyle.Invisible
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)

```

```

'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using

```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Get input Excel document from an embedded resource collection
    Stream excelStream =
        assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set the gridlines display style as Invisible
    settings.DisplayGridLines = GridLinesDisplayStyle.Invisible;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    Save(stream, "Output.pdf");
    excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (
        !(Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons"))
    )
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() { ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
}

```

```

}
if (stFile != null)
{
    Windows.Storage.Streams.IRandomAccessStream fileStream = await
    stFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream st = fileStream.AsStreamForWrite();
    st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
    st.Flush();
    st.Dispose();
    fileStream.Dispose();
}
stream.Dispose();
}
#endregion

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set the gridlines display style as Invisible
    settings.DisplayGridLines = GridLinesDisplayStyle.Invisible;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Saving the Excel to the MemoryStream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    //Set the position as '0'
    stream.Position = 0;
    //Download the PDF file in the browser
    FileStreamResult fileStreamResult = new FileStreamResult(stream,
    "application/pdf");
    fileStreamResult.FileName = "Output.pdf";
    return fileStreamResult;
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings

```

```

XlsIORendererSettings settings = new XlsIORendererSettings();
//Set the gridlines display style as Invisible
settings.DisplayGridLines = GridLinesDisplayStyle.Invisible;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}

```

Visible

Gridlines will be rendered in the output PDF document, when display style is selected as **Visible**. The following code snippet explains how to select **Visible** display style for gridlines.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Set the gridlines display style as Visible
    settings.DisplayGridLines = GridLinesDisplayStyle.Visible;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Set the gridlines display style as Visible
settings.DisplayGridLines = GridLinesDisplayStyle.Visible
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")

```


End Using**UWP**

```

//Excel To PDF conversion can be performed by referring .NET Standard assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Get input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set the gridlines display style as Visible
    settings.DisplayGridLines = GridLinesDisplayStyle.Visible;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    Save(stream, "Output.pdf");
    excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() { ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
        CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {

```

```

Windows.Storage.Streams.IRandomAccessStream fileStream = await
stFile.OpenAsync(FileAccessMode.ReadWrite);
Stream st = fileStream.AsStreamForWrite();
st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
st.Flush();
st.Dispose();
fileStream.Dispose();
}
stream.Dispose();
}
#endregion

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set the gridlines display style as Visible
    settings.DisplayGridLines = GridLinesDisplayStyle.Visible;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Saving the Excel to the MemoryStream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    //Set the position as '0'
    stream.Position = 0;
    //Download the PDF file in the browser
    FileStreamResult fileStreamResult = new FileStreamResult(stream,
    "application/pdf");
    fileStreamResult.FileNameDownloadName = "Output.pdf";
    return fileStreamResult;
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set the gridlines display style as Visible
    settings.DisplayGridLines = GridLinesDisplayStyle.Visible;
}

```

```
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}
```

Embed Fonts

The Excel content will be rendered improperly when Arial Unicode MS Font is missing in the machine. In this case, enabling the [EmbedFonts](#) property renders the content properly.

The following code snippet explains how to enable **EmbedFonts** property.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Enable EmbedFonts
    settings.EmbedFonts = true;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Enable EmbedFonts
settings.EmbedFonts = True
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using
```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Get input Excel document from an embedded resource collection
    Stream excelStream =
        assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Enable EmbedFonts
    settings.EmbedFonts = true;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    Save(stream, "Output.pdf");
    excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (
        (!Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons"))
    )
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() { ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
            stFile.OpenAsync(FileAccessMode.ReadWrite);
    }
}

```

```

Stream st = fileStream.AsStreamForWrite();
st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
st.Flush();
st.Dispose();
fileStream.Dispose();
}
stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Enable EmbedFonts
    settings.EmbedFonts = true;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Saving the Excel to the MemoryStream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    //Set the position as '0'
    stream.Position = 0;
    //Download the PDF file in the browser
    FileStreamResult fileStreamResult = new FileStreamResult(stream,
    "application/pdf");
    fileStreamResult.FileNameDownload = "Output.pdf";
    return fileStreamResult;
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Enable EmbedFonts
    settings.EmbedFonts = true;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
}

```

```
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}
```

Export Bookmarks

The default value of [ExportBookmarks](#) property is TRUE and hence bookmarks are exported to PDF document by default, in Excel to PDF conversion. Disabling this property skips the bookmarks in conversion.

The following code snippet explains how to disable [ExportBookmarks](#) property.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Disable ExportBookmarks
    settings.ExportBookmarks = false;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Disable ExportBookmarks
settings.ExportBookmarks = False
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using
```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Get input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("UWP.Sample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Disable ExportBookmarks
    settings.ExportBookmarks = false;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    Save(stream, "Output.pdf");
    excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
            ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
        stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
    }
}

```

```

st.Flush();
st.Dispose();
fileStream.Dispose();
}
stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Disable ExportBookmarks
    settings.ExportBookmarks = false;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Saving the Excel to the MemoryStream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    //Set the position as '0'
    stream.Position = 0;
    //Download the PDF file in the browser
    FileStreamResult fileStreamResult = new FileStreamResult(stream,
    "application/pdf");
    fileStreamResult.FileNameDownload = "Output.pdf";
    return fileStreamResult;
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Disable ExportBookmarks
    settings.ExportBookmarks = false;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream
}

```



```
MemoryStream stream = new MemoryStream();
document.Save(stream);
stream.Position = 0;
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}
```

Export Document Properties

Excel document properties will be exported to PDF document by default, in Excel to PDF conversion. This can be skipped by disabling the [ExportDocumentProperties](#).

The following complete code snippet explains how to disable ExportDocumentProperties.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Disable ExportDocumentProperties
    settings.ExportDocumentProperties = false;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Disable ExportDocumentProperties
settings.ExportDocumentProperties = False
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using
```

UWP

```
//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
```

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Get input Excel document from an embedded resource collection
    Stream excelStream =
        assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Disable ExportDocumentProperties
    settings.ExportDocumentProperties = false;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    Save(stream, "Output.pdf");
    excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (
        !(Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons"))
    )
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() { ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
            stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
}

```

```
stream.Dispose();  
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())  
{  
    IApplication application = excelEngine.Excel;  
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,  
    FileAccess.Read);  
    IWorkbook workbook = application.Workbooks.Open(excelStream);  
    //Initialize XlsIORendererSettings  
    XlsIORendererSettings settings = new XlsIORendererSettings();  
    //Disable ExportDocumentProperties  
    settings.ExportDocumentProperties = false;  
    //Initialize XlsIORenderer  
    XlsIORenderer renderer = new XlsIORenderer();  
    //Convert the Excel document to PDF with renderer settings  
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);  
    //Saving the Excel to the MemoryStream  
    MemoryStream stream = new MemoryStream();  
    document.Save(stream);  
    //Set the position as '0'  
    stream.Position = 0;  
    //Download the PDF file in the browser  
    FileStreamResult fileStreamResult = new FileStreamResult(stream,  
    "application/pdf");  
    fileStreamResult.FileNameDownload = "Output.pdf";  
    return fileStreamResult;  
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())  
{  
    IApplication application = excelEngine.Excel;  
    application.DefaultVersion = ExcelVersion.Excel2016;  
    //Gets assembly  
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;  
    //Gets input Excel document from an embedded resource collection  
    Stream excelStream =  
    assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");  
    IWorkbook workbook = application.Workbooks.Open(excelStream);  
    //Initialize XlsIORendererSettings  
    XlsIORendererSettings settings = new XlsIORendererSettings();  
    //Disable ExportDocumentProperties  
    settings.ExportDocumentProperties = false;  
    //Initialize XlsIORenderer  
    XlsIORenderer renderer = new XlsIORenderer();  
    //Convert the Excel document to PDF with renderer settings  
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);  
    //Save the PDF document to stream  
    MemoryStream stream = new MemoryStream();  
    document.Save(stream);  
    stream.Position = 0;
```

```
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}
```

Export Quality Image

The default value of [ExportQualityImage](#) is FALSE. Enabling this property exports quality image to PDF document.

The following complete code snippet explains how to enable ExportQualityImage property.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Enable ExportQualityImage
    settings.ExportQualityImage = true;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Enable ExportQualityImage
settings.ExportQualityImage = True
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using
```

UWP

```
//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
```

```

application.DefaultVersion = ExcelVersion.Excel2016;
//Get assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Get input Excel document from an embedded resource collection
Stream excelStream =
assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(excelStream);
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Enable ExportQualityImage
settings.ExportQualityImage = true;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
Save(stream, "Output.pdf");
excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
            ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
            stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
    stream.Dispose();
}

```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Enable ExportQualityImage
    settings.ExportQualityImage = true;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Saving the Excel to the MemoryStream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    //Set the position as '0'
    stream.Position = 0;
    //Download the PDF file in the browser
    FileStreamResult fileStreamResult = new FileStreamResult(stream,
    "application/pdf");
    fileStreamResult.FileNameDownload = "Output.pdf";
    return fileStreamResult;
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Enable ExportQualityImage
    settings.ExportQualityImage = true;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    stream.Position = 0;
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
    "application/pdf", stream);
}
```

Header Footer Option

Two properties available under [HeaderFooterOption](#) are ShowHeader and ShowFooter.

Show Header

Document header will be rendered to PDF document by default. This can be skipped by disabling the [ShowHeader](#) property.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Disable ShowHeader
    settings.HeaderFooterOption.ShowHeader = false;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Disable ShowHeader
settings.HeaderFooterOption.ShowHeader = False
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using
```

UWP

```
//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
```

```

Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Get input Excel document from an embedded resource collection
Stream excelStream =
assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(excelStream);
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Disable ShowHeader
settings.HeaderFooterOption.ShowHeader = false;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
Save(stream, "Output.pdf");
excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
    stream.Dispose();
}
}

```

ASP.NET CORE


```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Disable ShowHeader
    settings.HeaderFooterOption.ShowHeader = false;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Saving the Excel to the MemoryStream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    //Set the position as '0'
    stream.Position = 0;
    //Download the PDF file in the browser
    FileStreamResult fileStreamResult = new FileStreamResult(stream,
    "application/pdf");
    fileStreamResult.FileNameDownload = "Output.pdf";
    return fileStreamResult;
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Disable ShowHeader
    settings.HeaderFooterOption.ShowHeader = false;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    stream.Position = 0;
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
    "application/pdf", stream);
}
```

Show Footer

Document footer will be rendered to PDF document by default. This can be skipped by disabling the [ShowFooter](#) property.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Disable ShowFooter
    settings.HeaderFooterOption.ShowFooter = false;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Disable ShowFooter
settings.HeaderFooterOption.ShowFooter = False
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using
```

UWP

```
//Excel To PDF conversion can be performed by referring .NET Standard
//assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Get input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
}
```

```

//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Disable ShowFooter
settings.HeaderFooterOption.ShowFooter = false;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
Save(stream, "Output.pdf");
excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
            ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
            stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
    stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
}

```

```

FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
    FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(excelStream);
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Disable ShowFooter
settings.HeaderFooterOption.ShowFooter = false;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Saving the Excel to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF file in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
    "application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
        assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Disable ShowFooter
    settings.HeaderFooterOption.ShowFooter = false;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    stream.Position = 0;
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
        "application/pdf", stream);
}

```

Convert Blank Page

Blank pages in Excel document are rendered to PDF document by default, in Excel to PDF conversion. This blank page can be skipped by disabling the [IsConvertBlankPage](#) property.

The following code snippet explains this.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Disable IsConvertBlankPage
    settings.IsConvertBlankPage = false;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Disable IsConvertBlankPage
settings.IsConvertBlankPage = False
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using
```

UWP

```
//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Get input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
```

```

//Disable IsConvertBlankPage
settings.IsConvertBlankPage = false;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
Save(stream, "Output.pdf");
excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (
        !(Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons"))
    )
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() { ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
        stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
    stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
}

```

```
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Disable IsConvertBlankPage
settings.IsConvertBlankPage = false;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Saving the Excel to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF file in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
        assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Disable IsConvertBlankPage
    settings.IsConvertBlankPage = false;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    stream.Position = 0;
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}
```

Convert Blank Sheet

Blank worksheets in Excel document are rendered to PDF document by default, in Excel to PDF conversion. This blank worksheets can be skipped by disabling the [IsConvertBlankSheet](#) property.

The following code snippet explains this.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Disable IsConvertBlankSheet
    settings.IsConvertBlankSheet = false;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Disable IsConvertBlankSheet
settings.IsConvertBlankSheet = False
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using

```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Get input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Disable IsConvertBlankSheet
    settings.IsConvertBlankSheet = false;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
}

```



```

//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
Save(stream, "Output.pdf");
excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
            ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
        stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
    stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Disable IsConvertBlankSheet
    settings.IsConvertBlankSheet = false;
}

```

```
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Saving the Excel to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF file in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileName = "Output.pdf";
return fileStreamResult;
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Disable IsConvertBlankSheet
    settings.IsConvertBlankSheet = false;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    stream.Position = 0;
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
    "application/pdf", stream);
}
```

Layout Options

This property helps to select the layout option for the Excel document in Excel to PDF conversion. Five layout options available and maintained under [LayoutOptions](#) enumeration are **Automatic**, **CustomScaling**, **FitAllColumnsOnOnePage**, **FitAllRowsOnOnePage**, **FitSheetOnOnePage** and **NoScaling**.

Automatic

Selecting **Automatic** under LayoutOptions prints the worksheets at their actual size. It's default value is **NoScaling**. The following code snippet explains how to select the layout option as **Automatic**.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Set layout option as Automatic
    settings.LayoutOptions = LayoutOptions.Automatic;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Set layout option as Automatic
settings.LayoutOptions = LayoutOptions.Automatic
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using

```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Get input Excel document from an embedded resource collection
    Stream excelStream =
assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as Automatic
    settings.LayoutOptions = LayoutOptions.Automatic;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();

```

```

//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
Save(stream, "Output.pdf");
excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
            ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
        stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
    stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as Automatic
    settings.LayoutOptions = LayoutOptions.Automatic;
}

```

```
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Saving the Excel to the MemoryStream
MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF file in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileName = "Output.pdf";
return fileStreamResult;
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as Automatic
    settings.LayoutOptions = Syncfusion.XlsIORenderer.LayoutOptions.Automatic;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    stream.Position = 0;
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
    "application/pdf", stream);
}
```

Custom Scaling

Selecting **CustomScaling** under **LayoutOptions** prints the worksheets at specified scaling, i.e., zoom value set in under page setup. The following code snippet explains how to use this.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
```

```
//Initialize ExcelToPdfConverterSettings
ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
//Set layout option as CustomScaling
settings.LayoutOptions = LayoutOptions.CustomScaling;
//Load the Excel document into ExcelToPdfConverter
ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
//Convert the Excel document to PDF with converter settings
PdfDocument document = converter.Convert(settings);
//Save the PDF document
document.Save("Output.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Set layout option as CustomScaling
settings.LayoutOptions = LayoutOptions.CustomScaling
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using
```

UWP

```
//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
//Get assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Get input Excel document from an embedded resource collection
Stream excelStream =
assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(excelStream);
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Set layout option as CustomScaling
settings.LayoutOptions = LayoutOptions.CustomScaling;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
}
```

```

Save(stream, "Output.pdf");
excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
    stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as CustomScaling
    settings.LayoutOptions = LayoutOptions.CustomScaling;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Saving the Excel to the MemoryStream
}

```

```

MemoryStream stream = new MemoryStream();
document.Save(stream);
//Set the position as '0'
stream.Position = 0;
//Download the PDF file in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownloadName = "Output.pdf";
return fileStreamResult;
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as CustomScaling
    settings.LayoutOptions =
    Syncfusion.XlsIORenderer.LayoutOptions.CustomScaling;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    stream.Position = 0;
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}

```

[Fit All Columns On One Page](#)

Selecting **FitAllColumnsOnOnePage** under **LayoutOptions** renders all columns in Excel worksheet into single PDF page. The following code snippet explains how to select the layout option as **FitAllColumnsOnOnePage**.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Set layout option as FitAllColumnsOnOnePage
}

```



```

settings.LayoutOptions = LayoutOptions.FitAllColumnsOnOnePage;
//Load the Excel document into ExcelToPdfConverter
ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
//Convert the Excel document to PDF with converter settings
PdfDocument document = converter.Convert(settings);
//Save the PDF document
document.Save("Output.pdf");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Set layout option as FitAllColumnsOnOnePage
settings.LayoutOptions = LayoutOptions.FitAllColumnsOnOnePage
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using

```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
//Get assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Get input Excel document from an embedded resource collection
Stream excelStream =
assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(excelStream);
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Set layout option as FitAllColumnsOnOnePage
settings.LayoutOptions = LayoutOptions.FitAllColumnsOnOnePage;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
Save(stream, "Output.pdf");
excelStream.Dispose();
}

```

```

//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
            ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
            stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
    stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as FitAllColumnsOnOnePage
    settings.LayoutOptions = LayoutOptions.FitAllColumnsOnOnePage;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Saving the Excel to the MemoryStream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    //Set the position as '0'
}

```

```
stream.Position = 0;
//Download the PDF file in the browser
FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileNameDownload = "Output.pdf";
return fileStreamResult;
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as FitAllColumnsOnOnePage
    settings.LayoutOptions =
    Syncfusion.XlsIORenderer.LayoutOptions.FitAllColumnsOnOnePage;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    stream.Position = 0;
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}
```

[Fit All Rows On One Page](#)

Selecting **FitAllRowsOnOnePage** under LayoutOptions renders all rows in Excel worksheet into single PDF page. The following code snippet explains how to select the layout option as **FitAllRowsOnOnePage**.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Set layout option as FitAllRowsOnOnePage
    settings.LayoutOptions = LayoutOptions.FitAllRowsOnOnePage;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
```

```
//Convert the Excel document to PDF with converter settings
PdfDocument document = converter.Convert(settings);
//Save the PDF document
document.Save("Output.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Set layout option as FitAllRowsOnOnePage
settings.LayoutOptions = LayoutOptions.FitAllRowsOnOnePage
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using
```

UWP

```
//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
//Get assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Get input Excel document from an embedded resource collection
Stream excelStream =
assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(excelStream);
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Set layout option as FitAllRowsOnOnePage
settings.LayoutOptions = LayoutOptions.FitAllRowsOnOnePage;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
Save(stream, "Output.pdf");
excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
```

```

{
    stream.Position = 0;
    StorageFile stFile;
    if
    (
        !(Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons"))
    )
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() { ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename, CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
            stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
    stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open, FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as FitAllRowsOnOnePage
    settings.LayoutOptions = LayoutOptions.FitAllRowsOnOnePage;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Saving the Excel to the MemoryStream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    //Set the position as '0'
    stream.Position = 0;
    //Download the PDF file in the browser
}

```

```

FileStreamResult fileStreamResult = new FileStreamResult(stream,
"application/pdf");
fileStreamResult.FileName = "Output.pdf";
return fileStreamResult;
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as FitAllRowsOnOnePage
    settings.LayoutOptions =
    Syncfusion.XlsIORenderer.LayoutOptions.FitAllRowsOnOnePage;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    stream.Position = 0;
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
    "application/pdf", stream);
}

```

Fit Sheet On One Page

Selecting **FitSheetOnOnePage** under **LayoutOptions** renders every single worksheet in Excel workbook into, single PDF page. The following code snippet explains how to select the layout option as **FitSheetOnOnePage**.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Set layout option as FitSheetOnOnePage
    settings.LayoutOptions = LayoutOptions.FitSheetOnOnePage;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
}

```

```
//Save the PDF document
document.Save("Output.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Set layout option as FitSheetOnOnePage
settings.LayoutOptions = LayoutOptions.FitSheetOnOnePage
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using
```

UWP

```
//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
//Get assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Get input Excel document from an embedded resource collection
Stream excelStream =
assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(excelStream);
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Set layout option as FitSheetOnOnePage
settings.LayoutOptions = LayoutOptions.FitSheetOnOnePage;
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the Excel document to PDF with renderer settings
PdfDocument document = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
Save(stream, "Output.pdf");
excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
stream.Position = 0;
```

```

StorageFile stFile;
if
(
    !(Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons"))
)
{
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.DefaultFileExtension = ".pdf";
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() { ".pdf" });
    stFile = await savePicker.PickSaveFileAsync();
}
else
{
    StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
    stFile = await local.CreateFileAsync(filename,
        CreationCollisionOption.ReplaceExisting);
}
if (stFile != null)
{
    Windows.Storage.Streams.IRandomAccessStream fileStream = await
    stFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream st = fileStream.AsStreamForWrite();
    st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
    st.Flush();
    st.Dispose();
    fileStream.Dispose();
}
stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as FitSheetOnOnePage
    settings.LayoutOptions = LayoutOptions.FitSheetOnOnePage;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Saving the Excel to the MemoryStream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    //Set the position as '0'
    stream.Position = 0;
    //Download the PDF file in the browser
    FileStreamResult fileStreamResult = new FileStreamResult(stream,
        "application/pdf");
    fileStreamResult.FileNameDownload = "Output.pdf";
}

```



```
return fileStreamResult;
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
        assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as FitSheetOnOnePage
    settings.LayoutOptions =
        Syncfusion.XlsIORenderer.LayoutOptions.FitSheetOnOnePage;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    stream.Position = 0;
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
        "application/pdf", stream);
}
```

No Scaling

Selecting **NoScaling** under **LayoutOptions** prints the worksheets at their actual size. The following code snippet explains how to select the layout option as **NoScaling**.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Set layout option as NoScaling
    settings.LayoutOptions = LayoutOptions.NoScaling;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}
```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Set layout option as NoScaling
settings.LayoutOptions = LayoutOptions.NoScaling
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using

```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Get assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Get input Excel document from an embedded resource collection
    Stream excelStream =
    assembly.GetManifestResourceStream("UWPSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as NoScaling
    settings.LayoutOptions = LayoutOptions.NoScaling;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    Save(stream, "Output.pdf");
    excelStream.Dispose();
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))

```

```

{
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.DefaultFileExtension = ".pdf";
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
        ".pdf" });
    stFile = await savePicker.PickSaveFileAsync();
}
else
{
    StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
    stFile = await local.CreateFileAsync(filename,
        CreationCollisionOption.ReplaceExisting);
}
if (stFile != null)
{
    Windows.Storage.Streams.IRandomAccessStream fileStream = await
    stFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream st = fileStream.AsStreamForWrite();
    st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
    st.Flush();
    st.Dispose();
    fileStream.Dispose();
}
stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as NoScaling
    settings.LayoutOptions = LayoutOptions.NoScaling;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Saving the Excel to the MemoryStream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    //Set the position as '0'
    stream.Position = 0;
    //Download the PDF file in the browser
    FileStreamResult fileStreamResult = new FileStreamResult(stream,
        "application/pdf");
    fileStreamResult.FileNameDownload = "Output.pdf";
    return fileStreamResult;
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream =
        assembly.GetManifestResourceStream("XamarinSample.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set layout option as NoScaling
    settings.LayoutOptions = Syncfusion.XlsIORenderer.LayoutOptions.NoScaling;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    stream.Position = 0;
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
        "application/pdf", stream);
}

```

PDF Conformance Level

Excel to PDF converter settings allows you to set the PDF conformance level. Excel to PDF currently supports following PDF conformances.

- PDF/A-1b conformance
- PDF/X-1a conformance

Note: 1. To know more details about PDF conformance refer <https://help.syncfusion.com/file-formats/pdf/working-with-pdf-conformance>

2. Pdf_X1A2001 is not supported for NETStandard.

The following code illustrates how to set the [PdfConformanceLevel](#) while converting Excel workbook to PDF.

C#

```

using(ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
        ExcelOpenType.Automatic);
    //Open the Excel document to Convert
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Initialize Excel to PDF converter settings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
}

```

```
// Set the conformance for PDF/A-1b conversion
settings.PdfConformanceLevel = PdfConformanceLevel.Pdf_A1B;
//Convert Excel document into PDF document
PdfDocument pdfDocument = converter.Convert(settings);
//Save the PDF file
pdfDocument.Save("ExcelToPDF.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
'Open the Excel document to convert
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Initialize Excel to PDF converter settings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Set the conformance for PDF/A-1b conversion
settings.PdfConformanceLevel = PdfConformanceLevel.Pdf_A1B
'Convert Excel document into PDF document
Dim pdfDocument As PdfDocument = converter.Convert(settings)
'Save the PDF file
pdfDocument.Save("ExcelToPDF.pdf")
End Using
```

UWP

```
//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
#region Excel To PDF
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from an embedded resource collection
Stream excelStream = assembly.GetManifestResourceStream("ExcelToPDF.xlsx");
IWorkbook workbook = await application.Workbooks.OpenAsync(excelStream);
//Initialize XlsIO renderer.
XlsIORenderer renderer = new XlsIORenderer();
//Initialize XlsIO renderer settings
XlsIORendererSettings settings = new XlsIORendererSettings();
// Set the conformance for PDF/A-1b conversion
settings.PdfConformanceLevel = PdfConformanceLevel.Pdf_A1B;
//Convert Excel document into PDF document
PdfDocument pdfDocument = renderer.ConvertToPDF(workbook, settings);
//Save the PDF document to stream.
MemoryStream stream = new MemoryStream();
await pdfDocument.SaveAsync(stream);
Save(stream, "ExcelToPDF.pdf");
excelStream.Dispose();
stream.Dispose();
}
```

```

#endregion
//Save the workbook stream as a file.
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Sample";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
            ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
        stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
}
#endregion

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream = new FileStream("ExcelToPDF.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIO renderer.
    XlsIORenderer renderer = new XlsIORenderer();
    //Initialize XlsIO renderer settings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    // Set the conformance for PDF/A-1b conversion
    settings.PdfConformanceLevel = PdfConformanceLevel.Pdf_A1B;
    //Convert Excel document into PDF document
    PdfDocument pdfDocument = renderer.ConvertToPDF(workbook, settings);
    Stream stream = new FileStream("ExcelToPDF.pdf", FileMode.Create,
        FileAccess.ReadWrite);
    pdfDocument.Save(stream);
}

```

```
excelStream.Dispose();
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream excelStream = assembly.GetManifestResourceStream("ExcelToPDF.xlsx");
    IWorkbook workbook = application.Workbooks.Open(excelStream);
    //Initialize XlsIO renderer.
    XlsIORenderer renderer = new XlsIORenderer();
    //Initialize XlsIO renderer settings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    // Set the conformance for PDF/A-1b conversion
    settings.PdfConformanceLevel = PdfConformanceLevel.Pdf_A1B;
    //Convert Excel document into PDF document
    PdfDocument pdfDocument = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream.
    MemoryStream stream = new MemoryStream();
    pdfDocument.Save(stream);
    stream.Position = 0;
    //Save the stream into pdf file
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("ExcelToPDF.pdf", "application/pdf", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().Save("ExcelToPDF.pdf", "application/pdf", stream);
    }
    excelStream.Dispose();
    stream.Dispose();
}
```

Template Document

[TemplateDocument](#) property helps to store the PDF document.

The following complete code snippet explains how to convert and save two Excel documents into single PDF document using TemplateDocument property.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook1 = application.Workbooks.Open("Sample1.xlsx");
    IWorkbook workbook2 = application.Workbooks.Open("Sample2.xlsx");
```

```

//Load the Excel documents into ExcelToPdfConverter
ExcelToPdfConverter converter1 = new ExcelToPdfConverter(workbook1);
ExcelToPdfConverter converter2 = new ExcelToPdfConverter(workbook2);
//Convert the first Excel document to PDF
PdfDocument document = converter1.Convert();
//Initialize ExcelToPdfConverterSettings
ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
//Set the document as TemplateDocument
settings.TemplateDocument = document;
//Create a new PDF document and convert the second Excel document with settings
PdfDocument newDocument = new PdfDocument();
newDocument = converter2.Convert(settings);
//Save the new PDF Document
newDocument.Save("Output.pdf");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook1 As IWorkbook = application.Workbooks.Open("Sample1.xlsx")
Dim workbook2 As IWorkbook = application.Workbooks.Open("Sample2.xlsx")
'Load the Excel documents into ExcelToPdfConverter
Dim converter1 As ExcelToPdfConverter = New ExcelToPdfConverter(workbook1)
Dim converter2 As ExcelToPdfConverter = New ExcelToPdfConverter(workbook2)
'Convert the first Excel document to PDF
Dim document As PdfDocument = converter1.Convert
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Set the document as TemplateDocument
settings.TemplateDocument = document
'Create a new PDF document and convert the second Excel document with settings
Dim newDocument As PdfDocument = New PdfDocument
newDocument = converter2.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using

```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
//Get assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel documents from an embedded resource collection
Stream excelStream1 =
assembly.GetManifestResourceStream("UWPSample.Sample1.xlsx");
IWorkbook workbook1 = application.Workbooks.Open(excelStream1);

```



```

Stream excelStream2 =
assembly.GetManifestResourceStream("UWPSample.Sample2.xlsx");
IWorkbook workbook2 = application.Workbooks.Open(excelStream2);
//Initialize XlsIORenderer
XlsIORenderer renderer = new XlsIORenderer();
//Convert the first Excel document to PDF
PdfDocument document = renderer.ConvertToPDF(workbook1);
//Initialize XlsIORendererSettings
XlsIORendererSettings settings = new XlsIORendererSettings();
//Set the document as TemplateDocument
settings.TemplateDocument = document;
//Convert the second Excel document to PDF with renderer settings
PdfDocument newDocument = renderer.ConvertToPDF(workbook2, settings);
//Save the PDF document
MemoryStream stream = new MemoryStream();
document.Save(stream);
Save(stream, "Output.pdf");
excelStream1.Dispose();
excelStream2.Dispose();
}
//Save the workbook stream as a file.
#region Setting output location
async void Save(Stream stream, string filename)
{
stream.Position = 0;
StorageFile stFile;
if
(! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
{
FileSavePicker savePicker = new FileSavePicker();
savePicker.DefaultFileExtension = ".pdf";
savePicker.SuggestedFileName = "Sample";
savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
".pdf" });
stFile = await savePicker.PickSaveFileAsync();
}
else
{
StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
stFile = await local.CreateFileAsync(filename,
CreationCollisionOption.ReplaceExisting);
}
if (stFile != null)
{
Windows.Storage.Streams.IRandomAccessStream fileStream = await
stFile.OpenAsync(FileAccessMode.ReadWrite);
Stream st = fileStream.AsStreamForWrite();
st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
st.Flush();
st.Dispose();
fileStream.Dispose();
}
}
#endregion

```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    FileStream excelStream1 = new FileStream("Sample1.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook1 = application.Workbooks.Open(excelStream1);
    FileStream excelStream2 = new FileStream("Sample2.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook2 = application.Workbooks.Open(excelStream2);
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the first Excel document to PDF
    PdfDocument document = renderer.ConvertToPDF(workbook1);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set the document as TemplateDocument
    settings.TemplateDocument = document;
    //Convert the second Excel document to PDF with renderer settings
    PdfDocument newDocument = renderer.ConvertToPDF(workbook2, settings);
    //Saving the Excel to the MemoryStream
    MemoryStream stream = new MemoryStream();
    newDocument.Save(stream);
    //Set the position as '0'
    stream.Position = 0;
    //Download the PDF file in the browser
    FileStreamResult fileStreamResult = new FileStreamResult(stream,
    "application/pdf");
    fileStreamResult.FileNameDownload = "Output.pdf";
    return fileStreamResult;
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel documents from an embedded resource collection
    Stream excelStream1 =
    assembly.GetManifestResourceStream("XamarinSample.Sample1.xlsx");
    IWorkbook workbook1 = application.Workbooks.Open(excelStream1);
    Stream excelStream2 =
    assembly.GetManifestResourceStream("XamarinSample.Sample2.xlsx");
    IWorkbook workbook2 = application.Workbooks.Open(excelStream2);
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the first Excel document to PDF
    PdfDocument document = renderer.ConvertToPDF(workbook1);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Set the document as TemplateDocument
    settings.TemplateDocument = document;
}
```

```
//Convert the second Excel document to PDF with renderer settings
PdfDocument newDocument = renderer.ConvertToPDF(workbook2, settings);
//Save the PDF document to stream
MemoryStream stream = new MemoryStream();
newDocument.Save(stream);
stream.Position = 0;
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
"application/pdf", stream);
}
```

Throw When Excel File Is Empty

The default value of [ThrowWhenExcelFileIsEmpty](#) property is FALSE, and hence the empty Excel document will be converted to PDF without any exception. Enabling this property throws **ExcelToPdfConverterException**, saying that the Excel document is Empty.

The following code snippet explains this.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Initialize ExcelToPdfConverterSettings
    ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
    //Enabling ThrowWhenExcelFileIsEmpty throws exception
    settings.ThrowWhenExcelFileIsEmpty = true;
    //Load the Excel document into ExcelToPdfConverter
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    //Convert the Excel document to PDF with converter settings
    PdfDocument document = converter.Convert(settings);
    //Save the PDF document
    document.Save("Output.pdf");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Create(1)
'Initialize ExcelToPdfConverterSettings
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Enabling ThrowWhenExcelFileIsEmpty throws exception
settings.ThrowWhenExcelFileIsEmpty = True
'Load the Excel document into ExcelToPdfConverter
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Convert the Excel document to PDF with converter settings
Dim document As PdfDocument = converter.Convert(settings)
'Save the PDF document
document.Save("Output.pdf")
End Using
```

UWP

```

//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Enabling ThrowWhenExcelFileIsEmpty throws exception
    settings.ThrowWhenExcelFileIsEmpty = true;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    Save(stream, "Output.pdf");
}
//Save the PDF stream as a file
#region Setting output location
async void Save(Stream stream, string filename)
{
    stream.Position = 0;
    StorageFile stFile;
    if
    (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.U
I.Input.HardwareButtons")))
    {
        FileSavePicker savePicker = new FileSavePicker();
        savePicker.DefaultFileExtension = ".pdf";
        savePicker.SuggestedFileName = "Output";
        savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() {
            ".pdf" });
        stFile = await savePicker.PickSaveFileAsync();
    }
    else
    {
        StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
        stFile = await local.CreateFileAsync(filename,
            CreationCollisionOption.ReplaceExisting);
    }
    if (stFile != null)
    {
        Windows.Storage.Streams.IRandomAccessStream fileStream = await
        stFile.OpenAsync(FileAccessMode.ReadWrite);
        Stream st = fileStream.AsStreamForWrite();
        st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
        st.Flush();
        st.Dispose();
        fileStream.Dispose();
    }
    stream.Dispose();
}
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Enabling ThrowWhenExcelFileIsEmpty throws exception
    settings.ThrowWhenExcelFileIsEmpty = true;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Saving the Excel to the MemoryStream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    //Set the position as '0'
    stream.Position = 0;
    //Download the PDF file in the browser
    FileStreamResult fileStreamResult = new FileStreamResult(stream,
        "application/pdf");
    fileStreamResult.FileName = "Output.pdf";
    return fileStreamResult;
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Initialize XlsIORendererSettings
    XlsIORendererSettings settings = new XlsIORendererSettings();
    //Enabling ThrowWhenExcelFileIsEmpty throws exception
    settings.ThrowWhenExcelFileIsEmpty = true;
    //Initialize XlsIORenderer
    XlsIORenderer renderer = new XlsIORenderer();
    //Convert the Excel document to PDF with renderer settings
    PdfDocument document = renderer.ConvertToPDF(workbook, settings);
    //Save the PDF document to stream
    MemoryStream stream = new MemoryStream();
    document.Save(stream);
    stream.Position = 0;
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.pdf",
        "application/pdf", stream);
}

```

Capture Warnings in Excel-to-PDF Conversion

XlsIO intentionally skips unsupported elements and substitutes unsupported fonts. The elements that were not converted and the fonts that were intentionally substituted can be raised as warnings, to decide whether to proceed the conversion with the warnings or to stop the conversion.

It is recommended to implement `IWarning` interface in a supporting class. The interface holds the properties,

Type – the element that failed to convert **Description** – the description of the failed element

In addition, a decision to continue the conversion process can be done here by setting boolean value to the property `Cancel`. If `Cancel` is set to TRUE the conversion cancels, else the conversion continues.

The following code snippet shows how to capture warnings during Excel-to-PDF conversion.

C#

```
using Syncfusion.ExcelToPdfConverter;
using Syncfusion.Pdf;
using Syncfusion.XlsIO;
namespace CaptureWarnings
{
    class Program
    {
        static void Main(string[] args)
        {
            using(ExcelEngine excelEngine = new ExcelEngine())
            {
                IApplication application = excelEngine.Excel;
                application.DefaultVersion = ExcelVersion.Excel2013;
                IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
                    ExcelOpenType.Automatic);
                //Open the Excel document to convert.
                ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
                //Initialize warning class to capture warnings during the conversion.
                Warning warning = new Warning();
                //Initialize Excel-to-PDF converter settings.
                ExcelToPdfConverterSettings settings = new ExcelToPdfConverterSettings();
                //Set the warning class that is implemented.
                settings.Warning = warning;
                //Convert Excel document into PDF document.
                PdfDocument pdfDocument = converter.Convert(settings);
                //If conversion process canceled null returned.
                if(pdfDocument != null)
                {
                    //Save the PDF file.
                    pdfDocument.Save("ExcelToPDF.pdf");
                }
            }
        }
        /// <summary>
        /// A supporting class that implements IWarning.
        /// </summary>
        public class Warning : IWarning
        {
            public void ShowWarning(WarningInfo warning)
            {
                //Cancel the conversion process if the warning type is conditional
                //formatting.
                if (warning.Type == WarningType.ConditionalFormatting)
                {
                    Cancel = true;
                }
                //To view or log the warning, you can make use of warning.Description.
            }
        }
    }
}
```

```
public bool Cancel { get; set; }
}
```

VB.NET

```
Imports Syncfusion.ExcelToPdfConverter
Imports Syncfusion.Pdf
Imports Syncfusion.XlsIO
Namespace CaptureWarnings
Class Program
Private Shared Sub Main(ByVal args As String())
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
'Open the Excel document to convert.
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
'Initialize warning class to capture warnings during the conversion.
Dim warning As Warning = New Warning()
'Initialize Excel-to-PDF converter settings.
Dim settings As ExcelToPdfConverterSettings = New
ExcelToPdfConverterSettings()
'Set the warning class that is implemented.
settings.Warning = warning
'Convert Excel document into PDF document.
Dim pdfDocument As PdfDocument = converter.Convert(settings)
'If conversion process canceled null returned.
If pdfDocument IsNot Nothing Then
'Save the PDF file.
pdfDocument.Save("ExcelToPDF.pdf")
End Using
End Sub
End Class
''' <summary>
''' A supporting class that implements IWarning.
''' </summary>
Public Class Warning
Inherits IWarning
Public Sub ShowWarning(ByVal warning As WarningInfo)
'Cancel the conversion process if the warning type is conditional formatting.
If warning.Type = WarningType.ConditionalFormatting Then Cancel = True
'To view or log the warning, you can make use of warning.Description.
End Sub
Public Property Cancel As Boolean
End Class
End Namespace
```

UWP

```
//Excel To PDF conversion can be performed by referring .NET Standard
assemblies in UWP platform
using Syncfusion.XlsIORenderer;
using Syncfusion.Pdf;
using Syncfusion.XlsIO;
```

```

namespace CaptureWarnings
{
    class Program
    {
        void Button_Click(Object sender, EventArgs args)
        {
            using (ExcelEngine excelEngine = new ExcelEngine())
            {
                IApplication application = excelEngine.Excel;
                //Gets assembly
                Assembly assembly = typeof(App).GetTypeInfo().Assembly;
                //Gets input Excel document from an embedded resource collection
                Stream excelStream = assembly.GetManifestResourceStream("Sample.xlsx");
                IWorkbook workbook = await application.Workbooks.OpenAsync(excelStream);
                //Initialize warning class to capture warnings during the conversion.
                Warning warning = new Warning();
                //Initialize XlsIO renderer.
                XlsIORenderer renderer = new XlsIORenderer();
                //Initialize XlsIO renderer settings.
                XlsIORendererSettings settings = new XlsIORendererSettings();
                //Set the warning class that is implemented.
                settings.Warning = warning;
                //Convert Excel document into PDF document.
                PdfDocument pdfDocument = renderer.ConvertToPDF(workbook, settings);
                //If conversion process canceled null returned.
                if(pdfDocument != null)
                {
                    //Save the PDF document to stream.
                    MemoryStream stream = new MemoryStream();
                    await pdfDocument.SaveAsync(stream);
                    Save(stream, "ExcelToPDF.pdf");
                    stream.Dispose();
                }
                excelStream.Dispose();
            }
        }
        //Save the workbook stream as a file.
        #region Setting output location
        async void Save(Stream stream, string filename)
        {
            stream.Position = 0;
            StorageFile stFile;
            if
            (! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons")))
            {
                FileSavePicker savePicker = new FileSavePicker();
                savePicker.DefaultFileExtension = ".pdf";
                savePicker.SuggestedFileName = "Sample";
                savePicker.FileTypeChoices.Add("Adobe PDF Document", new List<string>() { ".pdf" });
                stFile = await savePicker.PickSaveFileAsync();
            }
            else
            {
                StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
            }
        }
    }
}

```



```

stFile = await local.CreateFileAsync(filename,
CreationCollisionOption.ReplaceExisting);
}
if (stFile != null)
{
Windows.Storage.Streams.IRandomAccessStream fileStream = await
stFile.OpenAsync(FileAccessMode.ReadWrite);
Stream st = fileStream.AsStreamForWrite();
st.Write((stream as MemoryStream).ToArray(), 0, (int)stream.Length);
st.Flush();
st.Dispose();
fileStream.Dispose();
}
}
}
/// <summary>
/// A supporting class that implements IWarning.
/// </summary>
public class Warning : IWarning
{
public void ShowWarning(WarningInfo warning)
{
//Cancel the converion process if the warning type is conditional
formatting.
if (warning.Type == WarningType.ConditionalFormatting)
Cancel = true;
//To view or log the warning, you can make use of warning.Description.
}
public bool Cancel { get; set; }
}
}

```

ASP.NET CORE

```

using Syncfusion.XlsIORenderer;
using Syncfusion.Pdf;
using Syncfusion.XlsIO;
namespace CaptureWarnings
{
class Program
{
static void Main(string[] args)
{
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Open the Excel document to convert.
FileStream excelStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read);
IWorkbook workbook = application.Workbooks.Open(excelStream);
//Initialize warning class to capture warnings during the conversion.
Warning warning = new Warning();
//Initialize XlsIO renderer.
XlsIORenderer renderer = new XlsIORenderer();
//Initialize XlsIO renderer settings.
XlsIORendererSettings settings = new XlsIORendererSettings();

```

```

//Set the warning class that is implemented.
settings.Warning = warning;
//Convert Excel document into PDF document.
PdfDocument pdfDocument = renderer.ConvertToPDF(workbook, settings);
//If conversion process canceled null returned.
if(pdfDocument != null)
{
    //Save the PDF file.
    Stream stream = new FileStream("ExcelToPDF.pdf", FileMode.Create,
    FileAccess.ReadWrite);
    pdfDocument.Save(stream);
    stream.Dispose();
}
excelStream.Dispose();
}
}
}
/// <summary>
/// A supporting class that implements IWarning.
/// </summary>
public class Warning : IWarning
{
    public void ShowWarning(WarningInfo warning)
    {
        //Cancel the conversion process if the warning type is conditional
        formatting.
        if (warning.Type == WarningType.ConditionalFormatting)
            Cancel = true;
        //To view or log the warning, you can make use of warning.Description.
    }
    public bool Cancel { get; set; }
}
}

```

XAMARIN

```

using Syncfusion.XlsIORenderer;
using Syncfusion.Pdf;
using Syncfusion.XlsIO;
namespace CaptureWarnings
{
    class Program
    {
        void Button_Click(Object sender, EventArgs args)
        {
            using (ExcelEngine excelEngine = new ExcelEngine())
            {
                IApplication application = excelEngine.Excel;
                //Gets assembly
                Assembly assembly = typeof(App).GetTypeInfo().Assembly;
                //Gets input Excel document from an embedded resource collection
                Stream excelStream = assembly.GetManifestResourceStream("Sample.xlsx");
                IWorkbook workbook = application.Workbooks.Open(excelStream);
                //Initialize warning class to capture warnings during the conversion.
                Warning warning = new Warning();
                //Initialize XlsIO renderer.
            }
        }
    }
}

```

```

XlsIORenderer renderer = new XlsIORenderer();
//Initialize XlsIO renderer settings.
XlsIORendererSettings settings = new XlsIORendererSettings();
//Set the warning class that is implemented.
settings.Warning = warning;
//Convert Excel document into PDF document
PdfDocument pdfDocument = renderer.ConvertToPDF(workbook, settings);
//If conversion process canceled null returned.
if(pdfDocument != null)
{
    //Save the PDF document to stream.
    MemoryStream stream = new MemoryStream();
    pdfDocument.Save(stream);
    stream.Position = 0;
    //Save the stream into pdf file
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().Save("ExcelToPDF.pdf", "application/pdf", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().Save("ExcelToPDF.pdf", "application/pdf", stream);
    }
    stream.Dispose();
}
excelStream.Dispose();
}
}
}
/// <summary>
/// A supporting class that implements IWarning.
/// </summary>
public class Warning : IWarning
{
    public void ShowWarning(WarningInfo warning)
    {
        //Cancel the conversion process if the warning type is conditional
        //formatting.
        if (warning.Type == WarningType.ConditionalFormatting)
            Cancel = true;
        //To view or log the warning, you can make use of warning.Description.
    }
    public bool Cancel { get; set; }
}
}

```

Worksheet to Image conversion

Assemblies Required

Refer to the following links for assemblies/nuget packages required based on platforms to convert the worksheet to image.

- [Assemblies Information](#)
- [NuGet Information](#)

Note: Worksheet to Image conversion works proper in Blazor server-side alone and not in client-side.

Convert as bitmap

The following code shows how to convert the specified range of rows and columns in the worksheet to bitmap.

C#

```
// Convert as bitmap  
Image image = sheet.ConvertToImage(1, 1, 10, 20);  
image.Save("Sample.png", ImageFormat.Png);
```

VB.NET

```
'Convert as bitmap  
Dim image As Image = sheet.ConvertToImage(1, 1, 10, 20)  
image.Save("Sample.png", ImageFormat.Png)
```

UWP

```
//Worksheet To Image conversion can be performed by referring .NET Standard  
assemblies in UWP platform.  
// Initialize XlsIORenderer  
application.XlsIORenderer = new XlsIORenderer();  
//Create a new memory stream to save the image  
Stream stream = new MemoryStream();  
//Convert worksheet to image and save it to stream.  
worksheet.ConvertToImage(1, 1, 10, 20, stream);
```

ASP.NET CORE

```
// Initialize XlsIORenderer  
application.XlsIORenderer = new XlsIORenderer();  
//Create a new memory stream to save the image  
Stream stream = new MemoryStream();  
//Convert worksheet to image and save it to stream.  
worksheet.ConvertToImage(1, 1, 10, 20, stream);
```

XAMARIN

```
// Initialize XlsIORenderer  
application.XlsIORenderer = new XlsIORenderer();  
//Create a new memory stream to save the image  
Stream stream = new MemoryStream();  
//Convert worksheet to image and save it to stream.  
worksheet.ConvertToImage(1, 1, 10, 20, stream);
```

Save as stream

The following code snippet shows how to save a sheet as stream.

C#

```
// Converts and save as stream
MemoryStream stream = new MemoryStream();
sheet.ConvertToImage(1, 1, 10, 20, ImageType.Metafile, stream);
```

VB.NET

```
'Converts and save as stream
Dim stream As MemoryStream = New MemoryStream()
sheet.ConvertToImage(1, 1, 10, 20, ImageType.Metafile, stream)
```

UWP

```
//Worksheet To Image conversion can be performed by referring .NET Standard
//assemblies in UWP platform.
// Initialize XlsIORenderer
application.XlsIORenderer = new XlsIORenderer();
//Create a new memory stream to save the image
Stream stream = new MemoryStream();
//Convert worksheet to image and save it to stream.
worksheet.ConvertToImage(1, 1, 10, 20, stream);
```

ASP.NET CORE

```
// Initialize XlsIORenderer
application.XlsIORenderer = new XlsIORenderer();
// Converts and save as stream
MemoryStream stream = new MemoryStream();
sheet.ConvertToImage(1, 1, 10, 20, stream);
```

XAMARIN

```
// Initialize XlsIORenderer
application.XlsIORenderer = new XlsIORenderer();
// Converts and save as stream
MemoryStream stream = new MemoryStream();
sheet.ConvertToImage(1, 1, 10, 20, stream);
```

The complete code snippet of the previous options is shown as follows.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Convert as bitmap
    Image image = sheet.ConvertToImage(1, 1, 10, 20);
    image.Save("Sample.png", ImageFormat.Png);
    //Converts and save as stream
    MemoryStream stream = new MemoryStream();
```

```

sheet.ConvertToImage(1, 1, 10, 20, ImageType.Metafile, stream);
//Save the workbook to disk
workbook.SaveAs("Sample.xlsx");
//No exception will be thrown if there are unsaved workbooks
excelEngine.ThrowNotSavedOnDestroy = false;
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Convert as bitmap
Dim image As Image = worksheet.ConvertToImage(1, 1, 10, 20)
image.Save("Sample.png", ImageFormat.Png)
'Converts and save as stream
Dim stream As MemoryStream = New MemoryStream()
worksheet.ConvertToImage(1, 1, 10, 20, ImageType.Metafile, stream)
'Save the workbook to disk
workbook.SaveAs("Sample.xlsx")
'No exception will be thrown if there are unsaved workbooks.
excelEngine.ThrowNotSavedOnDestroy = False
End Using

```

UWP

```

//Worksheet To Image conversion can be performed by referring .NET Standard assemblies in UWP platform.
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from an embedded resource collection
Stream inputStream =
assembly.GetManifestResourceStream("WorksheetToImage.Sample.xlsx");
IWorkbook workbook = application.Workbooks.Open(inputStream,
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
//Initialize XlsIORenderer
application.XlsIORenderer = new XlsIORenderer();
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Image Files", new List<string>() {
".png", ".jpeg", ".jpg" });
//Creates a storage file from the FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Converts and save to stream
var file = await storageFile.OpenAsync(FileAccessMode.ReadWrite);
Stream stream = file.AsStreamForWrite();

```

```

sheet.ConvertToImage(1, 1, 10, 20, stream);
await file.FlushAsync();
stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook =
        application.Workbooks.Open(File.OpenRead("Sample.xlsx"),
            ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Initialize XlsIORenderer
    application.XlsIORenderer = new XlsIORenderer();
    //Converts and save as stream
    MemoryStream stream = new MemoryStream();
    sheet.ConvertToImage(1, 1, 10, 20, stream);
    //Close and Dispose
    workbook.Close();
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream =
        assembly.GetManifestResourceStream("WorksheetToImage.Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet sheet = workbook.Worksheets[0];
    //Initializing XlsIORenderer
    application.XlsIORenderer = new XlsIORenderer();
    //Converts and save to stream.
    MemoryStream stream = new MemoryStream();
    sheet.ConvertToImage(1, 1, 10, 20, stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
        DependencyService.Get<ISaveWindowsPhone>()
            .SaveAndView("Test.png", "image/png", stream);
    else
        DependencyService.Get<ISave>().SaveAndView("Test.png", "image/png", stream);
}

```

- Note:** 1. Instance of XlsIORenderer class is mandatory to convert the worksheet to image using .NET Standard assemblies.
2. In .NET Standard, the Image format and quality can be specified using the ExportImageOptions class. By default the ImageFormat is set to PNG and ScalingMode is set to Best.
3. Worksheet to image conversion is supported from .NET Framework 2.0 and .NET Standard 1.4 onwards.

Non-Supported Features:

- Subscript/Superscript
- Shrink to fit
- Shapes (except TextBox shape and Image)
- Complex conditional formatting
- Gradient fill is partially supported

Chart to image conversion

Refer to the following links for assemblies/nuget packages required based on platforms to convert the chart to image.

- [Assemblies Information](#)
- [NuGet Information](#)

The following code snippet shows how to convert an Excel chart to an image using the **ExcelChartToImageConverter** class.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    application.ChartToImageConverter = new ChartToImageConverter();
    application.ChartToImageConverter.ScalingMode = ScalingMode.Best;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    IChart chart = worksheet.Charts[0];
    //Creating the memory stream for chart image
    MemoryStream stream = new MemoryStream();
    //Saving the chart as image
    chart.SaveAsImage(stream);
    Image image = Image.FromStream(stream);
    //Saving image stream to file
    image.Save("Output.png");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
```



```

Dim ChartToImageConverter As chartToImageConverter = New
ChartToImageConverter()
application.ChartToImageConverter = ChartToImageConverter
application.ChartToImageConverter.ScalingMode = ScalingMode.Best
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
Dim chart As IChart = worksheet.Charts(0)
'Creating the memory stream for chart image
Dim stream As New MemoryStream()
'Saving the chart as image
chart.SaveAsImage(stream)
Dim image As Image = Image.FromStream(stream)
'Saving image stream to file
image.Save("Output.png")
End Using

```

UWP

```

//Chart To Image conversion can be performed by referring .NET Standard
assemblies in UWP platform.
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Initializing XlsIORenderer
    application.XlsIORenderer = new XlsIORenderer();
    //Set converter chart image format to PNG
    application.XlsIORenderer.ChartRenderingOptions.ImageFormat =
    ExportImageFormat.Png;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream = assembly.GetManifestResourceStream("Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream,
    ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    IChart chart = worksheet.Charts[0];
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Image Files", new List<string>() {
    ".png", ".jpeg", ".jpg" });
    //Creates a storage file from the FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Converts and save to stream
    var file = await storageFile.OpenAsync(FileAccessMode.ReadWrite);
    Stream stream = file.AsStreamForWrite();
    chart.SaveAsImage(stream);
    await file.FlushAsync();
    stream.Dispose();
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())

```

```

{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    // Initialize XlsIORenderer
    application.XlsIORenderer = new XlsIORenderer();
    //Set converter chart image format to PNG
    application.XlsIORenderer.ChartRenderingOptions.ImageFormat =
    ExportImageFormat.Png;
    IWorkbook workbook =
    application.Workbooks.Open(File.OpenRead("Sample.xlsx"),
    ExcelOpenType.Automatic);
    IWorksheet worksheet = workbook.Worksheets[0];
    IChart chart = worksheet.Charts[0];
    //Creating the memory stream for chart image
    MemoryStream stream = new MemoryStream();
    //Saving the chart as image
    chart.SaveAsImage(stream);
    //Close and Dispose
    workbook.Close();
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    //Initializing XlsIORenderer
    application.XlsIORenderer = new XlsIORenderer();
    //Set converter chart image format to PNG
    application.XlsIORenderer.ChartRenderingOptions.ImageFormat =
    ExportImageFormat.Png;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from an embedded resource collection
    Stream inputStream = assembly.GetManifestResourceStream("Sample.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    IChart chart = worksheet.Charts[0];
    //Creating the memory stream for chart image
    MemoryStream stream = new MemoryStream();
    //Saving the chart as image
    chart.SaveAsImage(stream);
    stream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
        DependencyService.Get<ISaveWindowsPhone>().
        SaveAndView("Test.png", "image/png", stream);
    else
        DependencyService.Get<ISave>().SaveAndView("Test.png", "image/png", stream);
}

```

Note: 1. Instance of XlsIORenderer class is mandatory to convert the chart to image using .NET Standard 2.0 assemblies.

2. In .NET Standard, the Image format and quality can be specified using the ChartRenderingOptions property of XlsIORenderer class. By default the ImageFormat for chart is set to JPEG and ScalingMode is set to Best.

3. Chart conversion to image and PDF are supported from .NET Framework 4.0 and .NET Standard 2.0 onwards

4. Chart to Image conversion also works proper in Blazor server-side alone and not in client-side.

Supported chart types

XlsIO supports the following chart types in image conversion.

Chart Type	Chart Subtypes
Area	<i>Area</i> <i>AreaStacked</i> <i>AreaStacked100</i> <i>Area3D</i>
Bar	<i>BarClustered</i> <i>BarStacked</i> <i>BarStacked100</i> <i>BarClustered3D</i> <i>BarStacked3D</i> <i>BarStacked100_3D</i>
Bubble	Bubble
Column	<i>ColumnClustered</i> <i>ColumnStacked</i> <i>ColumnStacked100</i> <i>Column3D</i> <i>ColumnClustered3D</i> <i>ColumnStacked3D</i> <i>* ColumnStacked1003D</i>
Doughnut	<i>Doughnut</i> <i>Doughnut_Exploded</i>
Line	<i>Line</i> <i>LineMarkers</i> <i>* Line3D</i>
Pie	<i>Pie</i> <i>PieExploded</i> <i>Pie3D</i> <i>PieExploded3D</i>

Radar	<i>Radar</i> <i>RadarFilled</i> <i>* RadarMarkers</i>
Scatter	<i>ScatterLine</i> <i>ScatterLineMarkers</i> <i>ScatterMarkers</i> <i>ScatterSmoothedLine</i> <i>* ScatterSmoothedLine_Markers</i>
Stock	<i>StockHighLowClose</i> <i>StockOpenHighLowClose</i>
Excel 2016 Charts	<i>Funnel</i> <i>Waterfall</i> <i>Histogram</i> <i>Pareto</i>

Note: From the above supported chart types table, Waterfall and Line_3D charts are not supported in chart to image conversion in .NET Standard 2.0 onwards.

Supported chart elements

XlsIO supports the following chart elements in image conversion:

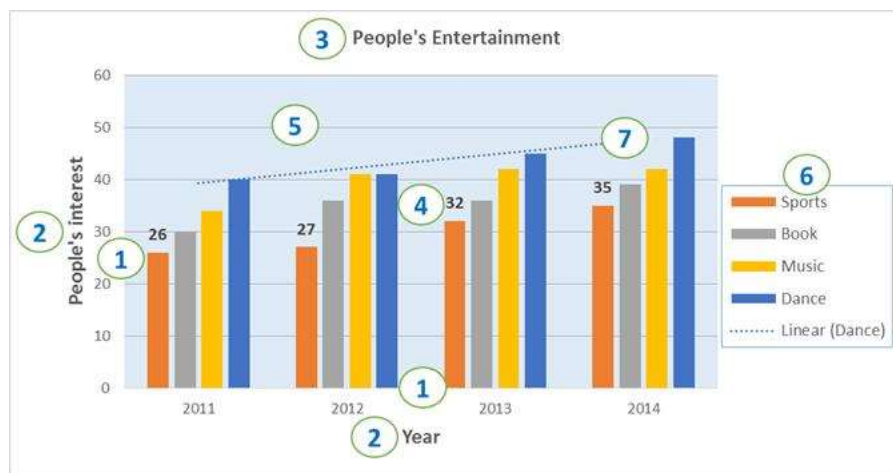


Chart Elements:

1. Axis
2. Axis titles
3. Chart title
4. Data labels
5. Grid lines
6. Legend
7. Trend line

Excel to ODS Conversion

The Open Document Format for Office Applications (ODF) is also known as OpenDocument. It is an XML-based file format for spreadsheets, charts, presentations, and word processing documents. It was developed with the aim of providing an open and XML-based file format specification for office applications. OpenOffice uses ODF format as its default document format. The OpenDocument Spreadsheet (ODS) is the file format for Excel documents. XlsIO supports conversion of XLS/XLSX documents to ODS.

Saving ODS in different platforms

The following code snippet illustrates the creation of an Excel file and exporting it to ODS format.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A1"].Text = "Month";
    worksheet.Range["B1"].Text = "Sales";
    worksheet.Range["A5"].Text = "Total";
    worksheet.Range["A2"].Text = "January";
    worksheet.Range["A3"].Text = "February";
    worksheet.AutofitColumn(1);
    worksheet.Range["B2"].Number = 68878;
    worksheet.Range["B3"].Number = 71550;
    worksheet.Range["B5"].Formula = "SUM(B2:B4)";
    //Comments
    IComment comment = worksheet.Range["B5"].AddComment();
    comment.RichText.Text = "This cell has formula.";
    IRichTextString richText = comment.RichText;
    IFont blueFont = workbook.CreateFont();
    blueFont.Color = ExcelKnownColors.Blue;
    richText.SetFont(0, 13, blueFont);
    IFont redFont = workbook.CreateFont();
    redFont.Color = ExcelKnownColors.Red;
    richText.SetFont(14, 20, redFont);
    //Formatting
    IStyle style = workbook.Styles.Add("Style1");
    style.Color = Color.DarkBlue;
    style.Font.Color = ExcelKnownColors.WhiteCustom;
    worksheet.Range["A1:B1"].CellStyleName = "Style1";
    worksheet.Range["A5:B5"].CellStyleName = "Style1";
    //Save in ODS format
    workbook.SaveAs("Output.ods");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
```

```

worksheet.Range("A1").Text = "Month"
worksheet.Range("B1").Text = "Sales"
worksheet.Range("A5").Text = "Total"
worksheet.Range("A2").Text = "January"
worksheet.Range("A3").Text = "February"
worksheet.AutofitColumn(1)
worksheet.Range("B2").Number = 68878
worksheet.Range("B3").Number = 71550
worksheet.Range("B5").Formula = "SUM(B2:B4) "
'Comments
Dim comment As IComment = worksheet.Range("B5").AddComment()
comment.RichText.Text = "This cell has formula."
Dim richText As IRichTextString = comment.RichText
Dim blueFont As IFont = workbook.CreateFont()
blueFont.Color = ExcelKnownColors.Blue
richText.SetFont(0, 13, blueFont)
Dim redFont As IFont = workbook.CreateFont()
redFont.Color = ExcelKnownColors.Red
richText.SetFont(14, 20, redFont)
'Formatting
Dim style As IStyle = workbook.Styles.Add("Style1")
style.Color = Color.DarkBlue
style.Font.Color = ExcelKnownColors.WhiteCustom
worksheet.Range("A1:B1").CellStyleName = "Style1"
worksheet.Range("A5:B5").CellStyleName = "Style1"
'Save in ODS format
workbook.SaveAs("Output.ods")
End Using

```

UWP

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A1"].Text = "Month";
    worksheet.Range["B1"].Text = "Sales";
    worksheet.Range["A5"].Text = "Total";
    worksheet.Range["A2"].Text = "January";
    worksheet.Range["A3"].Text = "February";
    worksheet.AutofitColumn(1);
    worksheet.Range["B2"].Number = 68878;
    worksheet.Range["B3"].Number = 71550;
    worksheet.Range["B5"].Formula = "SUM(B2:B4) ";
    //Comments
    IComment comment = worksheet.Range["B5"].AddComment();
    comment.RichText.Text = "This cell has formula.";
    IRichTextString richText = comment.RichText;
    IFont blueFont = workbook.CreateFont();
    blueFont.Color = ExcelKnownColors.Blue;
    richText.SetFont(0, 13, blueFont);
    IFont redFont = workbook.CreateFont();
    redFont.Color = ExcelKnownColors.Red;
    richText.SetFont(14, 20, redFont);
}

```

```

//Formatting
IStyle style = workbook.Styles.Add("Style1");
style.Color = Color.FromArgb(255, 72, 61, 139);
style.Font.Color = ExcelKnownColors.WhiteCustom;
worksheet.Range["A1:B1"].CellStyleName = "Style1";
worksheet.Range["A5:B5"].CellStyleName = "Style1";
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".ods"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile, ExcelSaveType.SaveAsODS);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
worksheet.Range["A1"].Text = "Month";
worksheet.Range["B1"].Text = "Sales";
worksheet.Range["A5"].Text = "Total";
worksheet.Range["A2"].Text = "January";
worksheet.Range["A3"].Text = "February";
worksheet.AutofitColumn(1);
worksheet.Range["B2"].Number = 68878;
worksheet.Range["B3"].Number = 71550;
worksheet.Range["B5"].Formula = "SUM(B2:B4)";
//Comments
IComment comment = worksheet.Range["B5"].AddComment();
comment.RichText.Text = "This cell has formula.";
IRichTextString richText = comment.RichText;
IFont blueFont = workbook.CreateFont();
blueFont.Color = ExcelKnownColors.Blue;
richText.SetFont(0, 13, blueFont);
IFont redFont = workbook.CreateFont();
redFont.Color = ExcelKnownColors.Red;
richText.SetFont(14, 20, redFont);
//Formatting
IStyle style = workbook.Styles.Add("Style1");
style.Color = Color.DarkBlue;
style.Font.Color = ExcelKnownColors.WhiteCustom;
worksheet.Range["A1:B1"].CellStyleName = "Style1";
worksheet.Range["A5:B5"].CellStyleName = "Style1";
//Saving the workbook as stream
FileStream stream = new FileStream("Output.ods", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream, ExcelSaveType.SaveAsODS);
stream.Dispose();
}

```

```
}

```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    worksheet.Range["A1"].Text = "Month";
    worksheet.Range["B1"].Text = "Sales";
    worksheet.Range["A5"].Text = "Total";
    worksheet.Range["A2"].Text = "January";
    worksheet.Range["A3"].Text = "February";
    worksheet.AutofitColumn(1);
    worksheet.Range["B2"].Number = 68878;
    worksheet.Range["B3"].Number = 71550;
    worksheet.Range["B5"].Formula = "SUM(B2:B4) ";
    //Comments
    IComment comment = worksheet.Range["B5"].AddComment();
    comment.RichText.Text = "This cell has formula.";
    IRichTextString richText = comment.RichText;
    IFont blueFont = workbook.CreateFont();
    blueFont.Color = ExcelKnownColors.Blue;
    richText.SetFont(0, 13, blueFont);
    IFont redFont = workbook.CreateFont();
    redFont.Color = ExcelKnownColors.Red;
    richText.SetFont(14, 20, redFont);
    //Formatting
    IStyle style = workbook.Styles.Add("Style1");
    style.Color = Syncfusion.Drawing.Color.DarkBlue;
    style.Font.Color = ExcelKnownColors.WhiteCustom;
    worksheet.Range["A1:B1"].CellStyleName = "Style1";
    worksheet.Range["A5:B5"].CellStyleName = "Style1";
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream, ExcelSaveType.SaveAsODS);
    string fileName = "Output.ods";
    outputStream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies between Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
            "application/msexcel", outputStream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
            "application/msexcel", outputStream);
    }
}
```


Supported and unsupported elements in ODS conversion

Category	Subcategory	Supported
Formatting	Font settings	Yes
	Alignments	Yes (Except indent)
	Number formatting	Yes (Partial)
	Border settings	Yes
	Fill settings	Yes
	RGB colors	Yes
	Cell gradient	No
	Cell styles	Yes
	Themes	No
	Conditional formatting	Planned
Hyperlinks	-	Yes
Cell Comments		Yes
Print	Page setup	Yes
	[Margin, Page size]	Yes
	Page breaks	No
	Background image	No
	Print settings [Print area, Print titles, Page order]	No
	Header/Footer	Planned
Formulas		Yes (Partial)
	Table Formulas	No
	Names	Yes (Partial)
Group & Outline		Yes
Settings	Window Settings	No

	Sheet/Book settings	No
Protection	Sheet Protection	No
	Encryption	N/A
Hide/Unhide rows/cols		Yes
Copy/Move worksheet		Yes
Image		No
Data Validation		No
Tables		Planned
PivotTable		No
Charts		Planned
Drawing		Planned
OLE Objects		No

Custom XML Support

When you embed XML data in a document, the data is named as custom XML part, which is used to store arbitrary XML data in the workbook.

Essential XlsIO supports the following functionalities with Custom XML:

- Adding CustomXmlPart to workbook
- Reading CustomXmlPart from workbook

Add Custom XML

Adding Custom XML part to workbook is achieved by using the **Add** method of **ICustomXmlPart** interface.

The following code snippet illustrates on how to add a Custom XML part.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding CustomXmlData to Workbook
    ICustomXmlPart customXmlPart = workbook.CustomXmlparts.Add("SD10003");
    //Add XmlData to CustomXmlPart
    byte[] xmlData = File.ReadAllBytes("Test.xml");
    customXmlPart.Data = xmlData;
    workbook.SaveAs("CustomXml.xlsx");
}
```

```
}

```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding CustomXmlData to Workbook
Dim customXmlPart As ICustomXmlPart = workbook.CustomXmlparts.Add("SD10003")
'Add XmlData to CustomXmlPart
Dim xmlData() As Byte = File.ReadAllBytes("Test.xml")
customXmlPart.Data = xmlData
workbook.SaveAs("CustomXml.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Gets assembly
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
//Gets input Excel document from embedded resource collection
Stream inputStream =
assembly.GetManifestResourceStream("CustomXml.Test.xml");
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Adding CustomXmlData to Workbook
ICustomXmlPart customXmlPart = workbook.CustomXmlparts.Add("SD10003");
//Add XmlData to CustomXmlPart
customXmlPart.Data = new byte[inputStream.Length];
inputStream.Read(customXmlPart.Data, 0, customXmlPart.Data.Length);
//Initializes FileSavePicker
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "CustomXml";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
```

```
//Adding CustomXmlData to Workbook
ICustomXmlPart customXmlPart = workbook.CustomXmlparts.Add("SD10003");
//Add XmlData to CustomXmlPart
byte[] xmlData = File.ReadAllBytes("Test.xml");
customXmlPart.Data = xmlData;
//Saving the workbook as stream
FileStream stream = new FileStream("CustomXml.xlsx", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("CustomXml.Test.xml");
    IWorkbook workbook = application.Workbooks.Create(1);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Adding CustomXmlData to Workbook
    ICustomXmlPart customXmlPart = workbook.CustomXmlparts.Add("SD10003");
    //Add XmlData to CustomXmlPart
    customXmlPart.Data = new byte[inputStream.Length];
    inputStream.Read(customXmlPart.Data, 0, customXmlPart.Data.Length);
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
    string fileName = "CustomXml.xlsx";
    outputStream.Position = 0;
    //Save the document as file and view the saved document
    //The operation in SaveAndView under Xamarin varies among Windows Phone,
    //Android, and iOS platforms. Refer to the xlsio/xamarin section for
    //respective code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
    TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName,
        "application/msexcel", outputStream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName,
        "application/msexcel", outputStream);
    }
}
```

Read Custom XML

Reading Custom XML part from workbook is achieved by using the **GetById** method of **ICustomXmlPart** interface. The following code snippet illustrates on how to read Custom XML parts from workbook.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Open("CustomXml.xlsx");
    IWorksheet worksheet = workbook.Worksheets[0];
    //Access CustomXmlPart from Workbook
    ICustomXmlPart customXmlPart = workbook.CustomXmlparts.GetById("SD10003");
    //Access XmlData from CustomXmlPart
    byte[] xmlData = customXmlPart.Data;
    System.Text.Encoding.Default.GetString(xmlData);
    workbook.SaveAs("CustomXml.xlsx");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("CustomXml.xlsx")
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Access CustomXmlPart from Workbook
Dim customXmlPart As ICustomXmlPart =
workbook.CustomXmlparts.GetById("SD10003")
'Access XmlData from CustomXmlPart
Dim xmlData As Byte() = customXmlPart.Data
System.Text.Encoding.Default.GetString(xmlData)
workbook.SaveAs("CustomXml.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("CustomXml.CustomXml.xlsx");
    IWorkbook workbook = await application.Workbooks.OpenAsync(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Access CustomXmlPart from Workbook
    ICustomXmlPart customXmlPart = workbook.CustomXmlparts.GetById("SD10003");
    //Access XmlData from CustomXmlPart
    byte[] xmlData = customXmlPart.Data;
    System.Text.Encoding.UTF8.GetString(xmlData);
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
}
```

```

savePicker.SuggestedFileName = "CustomXml";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
//Creates a storage file from FileSavePicker
StorageFile storageFile = await savePicker.PickSaveFileAsync();
//Saves changes to the specified storage file
await workbook.SaveAsAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    FileStream fileStream = new FileStream("CustomXml.xlsx", FileMode.Open,
    FileAccess.Read);
    IWorkbook workbook = application.Workbooks.Open(fileStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Access CustomXmlPart from Workbook
    ICustomXmlPart customXmlPart = workbook.CustomXmlparts.GetById("SD10003");
    //Access XmlData from CustomXmlPart
    byte[] xmlData = customXmlPart.Data;
    System.Text.Encoding.Default.GetString(xmlData);
    //Saving the workbook as stream
    FileStream stream = new FileStream("CustomXml1.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(stream);
    stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    //Gets assembly
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    //Gets input Excel document from embedded resource collection
    Stream inputStream =
    assembly.GetManifestResourceStream("CustomXml.CustomXml.xlsx");
    IWorkbook workbook = application.Workbooks.Open(inputStream);
    IWorksheet worksheet = workbook.Worksheets[0];
    //Access CustomXmlPart from Workbook
    ICustomXmlPart customXmlPart = workbook.CustomXmlparts.GetById("SD10003");
    //Access XmlData from CustomXmlPart
    byte[] xmlData = customXmlPart.Data;
    System.Text.Encoding.UTF8.GetString(xmlData, 0, xmlData.Length);
    //Saving the workbook as stream
    MemoryStream outputStream = new MemoryStream();
    workbook.SaveAs(outputStream);
    string fileName = "CustomXml.xlsx";
    outputStream.Position = 0;
    //Save the document as file and view the saved document
}

```

```
//The operation in SaveAndView under Xamarin varies among Windows Phone, Android, and iOS platforms. Refer to the xlsio/xamarin section for respective code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS == TargetPlatform.Windows)
{
    Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView(fileName, "application/msexcel", outputStream);
}
else
{
    Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView(fileName, "application/msexcel", outputStream);
}
}
```

Note: Custom XML cannot be modified when the file is saved in Excel 97-2003 (*.xls) format.

Custom XML can be created and modified when the file is saved in Excel 2007 and later versions (*.xlsx).

Migration from Office Automation to Syncfusion XlsIO

Why Syncfusion's XlsIO over Office Automation?

- Office automation can be extremely slow when deployed on a server. It is highly unstable when multiple Excel objects are instantiated on the server. However, XlsIO is designed to work under these circumstances and is a 100% native .NET component, so, it can easily handle concurrent requests from multiple users.
- XlsIO has a very intuitive and easy to use object model. You can create richly formatted spreadsheets in just a few minutes.
- Essential XlsIO has a royalty free licensing model compared to office automation, where the users are expected to have Microsoft Excel licenses when deployed on the server.

To migrate from Microsoft's Interop to Syncfusion's XlsIO due to its stunning performance, it is an easy way. If you don't know how to use the features, this section provides some important Excel features, which are listed below. When you click a topic, it navigates to the particular page. Each topic contains:

- Feature description
- Interop and XlsIO codes(both in C# and VB) for comparison and easy migration.

Topic links

- [Add Worksheets to Workbook](#)
- [Activate a Worksheet](#)
- [Protect Excel Workbook](#)
- [Unprotect Excel Workbook](#)
- [Create Pie Chart in Excel](#)
- [Use Formulas in Excel](#)
- [Create Named Range in Excel](#)
- [Add Comments in Excel](#)
- [Delete Comments in Excel](#)

- [Merge Cells in Excel](#)
- [Unmerge Cells in Excel](#)
- [Apply Borders in Excel](#)
- [Add Hyperlinks in Excel](#)
- [Hide Excel Worksheets](#)
- [Unhide Excel Worksheets](#)
- [Rotate Text in Cells](#)
- [Filter Excel Data](#)
- [Wrap Text in Excel](#)
- [Set Background for Excel Worksheet](#)

Supported Features by File Formats

The list of supported and non-supported Excel file format features in *Essential XlsIO* is given in the following table. *XLS* represents Excel 97 to 2003 format and *XLSX* represents Excel 2007 and above formats.

Feature	XLS			XLSX			XLS to XLSX
	Read	Write	Preserve	Read	Write	Preserve	
Document Properties	Yes	Yes	-	Yes	Yes	-	Yes
Font settings	Yes	Yes	-	Yes	Yes	-	Yes
Alignments	Yes	Yes	-	Yes	Yes	-	Yes
Number formatting	Yes	Yes	-	Yes	Yes	-	Yes
Border settings	Yes	Yes	-	Yes	Yes	-	Yes
Fill settings	Yes	Yes	-	Yes	Yes	-	Yes
Cell styles	Yes	Yes	-	Yes	Yes	-	Yes
Conditional formatting	Yes	Yes	-	Yes	Yes	-	Yes
Cell size[Row/column height/width, autofit]	Yes	Yes	-	Yes	Yes	-	Yes
Hide/Unhide rows/cols	Yes	Yes	-	Yes	Yes	-	Yes
Hide/Unhide worksheet	Yes	Yes	-	Yes	Yes	-	Yes
Copy/Move worksheet	Yes	Yes	-	Yes	Yes	-	-
Sheet protection	Yes	Yes	-	Yes	Yes	-	Yes

Workbook protection	Yes	Yes	-	Yes	Yes	-	Yes
Sheet format[sheet name, tab color]	Yes	Yes	-	Yes	Yes	-	Yes
Image	Yes	Yes	-	Yes	Yes	-	Yes
Charts	Yes	Yes	-	Yes	Yes	-	Yes
Hyperlinks	Yes	Yes	-	Yes	Yes	-	Yes
Header/Footer	Yes	Yes	-	Yes	Yes	-	Yes
Pivot tables	No	No	Yes	Yes	Yes	-	No
Pivot Chart	No	No	Yes	Yes	Yes	-	No
AutoShapes	No	No	Yes	Yes	Yes	-	No
Text Box	Yes	Yes	-	Yes	Yes	-	Yes
Check Box	Yes	Yes	-	Yes	Yes	-	Yes
Combo Box	Yes	Yes	-	Yes	Yes	-	Yes
Page setup	Yes	Yes	-	Yes	Yes	-	Yes
[Margin,origin,page size]							
Page breaks	Yes	Yes	-	Yes	Yes	-	Yes
Background image	Yes	Yes	-	Yes	Yes	-	Yes
Print settings[Print area,Print titles,page order]	Yes	Yes	-	Yes	Yes	-	Yes
Formulas	Yes	Yes	-	Yes	Yes	-	Yes
Calculation options	Yes	Yes	-	Yes	Yes	-	Yes
Names	Yes	Yes	-	Yes	Yes	-	Yes
Formula auditing [Ignore error]	Yes	Yes	-	Yes	Yes	-	Yes

Filter	Yes	Yes	-	Yes	Yes	-	Yes
Data validation	Yes	Yes	-	Yes	Yes	-	Yes
Custom XML	Yes	Yes	-	Yes	Yes	-	Yes
Collection Objects	Yes	Yes	-	Yes	Yes	-	Yes
Template marker	Yes	Yes	-	Yes	Yes	-	Yes
Outlines[group/ungroup, summary settings]	Yes	Yes	-	Yes	Yes	-	Yes
Comments	Partial	Partial	-	partial	partial	-	Yes
Freeze pane, split pane	Yes	Yes	-	Yes	Yes	-	Yes
View[Zoom, show/hide gridlines, show/hide headings], horizontal/vertical scroll bars	Yes	Yes	-	Yes	Yes	-	Yes
Macros	No	No	Yes	No	No	Yes	No
Encryption and Decryption	Yes	Yes	-	Yes	Yes	-	Yes
Track changes	No	No	No	No	No	No	No
Themes	-	-	-	Yes partial	No	-	-
Cell gradient	-	-	-	Yes	Yes	-	-
Conditional Formatting	Yes	Yes	-	Yes	Yes	-	Yes
Advanced CF [Icon Set, Data bars, Color scales, Specific Date]	-	-	-	Yes	Yes	-	-
Tables	No	No	No	Yes	Yes	-	No
RGB colors	Yes	Yes	Indexed color	Yes	Yes	-	Yes















OLE Objects	No	No	Yes (Full Trust Only)	Yes (Full Trust Only)	Yes (Full Trust Only)	-	No
-------------	----	----	-----------------------	-----------------------	-----------------------	---	----





Supported Features by Platforms













The list of various supported and non-supported Excel features of Essential XlsIO in different platforms is given in this section.











Supported Features by .NET Framework and .NET Standard







The list of supported and non-supported Excel features in .NET Framework and .NET Standard is given below.











Workbook Features	.NET Framework (WF, WPF, ASP.NET and ASP.NET MVC)	.NET Standard (UWP, .NET Core and Xamarin)
Create an Excel workbook from scratch or modify an existing workbook.		
Open an existing Excel workbook from file system or stream.		
Save Excel workbook to a local file, stream, or stream it to a client browser.		
Read worksheets on demand when Excel file contains multiple worksheets.		
Hide and unhide worksheets.		
Add or extract custom XML documents.		
Add or modify document properties.		









Import and Export	.NET Framework (WF, WPF, ASP.NET and ASP.NET MVC)	.NET Standard (UWP, .NET Core and Xamarin)
Import data from DataTable, DataColumn, DataView, Array to worksheet.		
Import data from collection objects to worksheet.		



Import data from nested collection objects to worksheet.		
Import data into a pre-formatted template using Template Markers.		
Importing large amounts of data with minimal memory consumption can be achieved using the import on save option.		
Export data from worksheet to datatable and collection objects.		
Export data from worksheet to nested class.		
Imports data from various external sources like Microsoft Access, SQL Server and Excel, which maintain the connection to refresh data dynamically.		

































Formulas	.NET Framework (WF, WPF, ASP.NET and ASP.NET MVC)	.NET Standard (UWP, .NET Core and Xamarin)
Read and write Excel formulas.		
Add or modify named ranges in workbook and worksheet levels.		
Performs calculation for a range of cells using named ranges.		
Apply or modify Excel formula auditing settings.		
Enable automatic and manual calculation options.		

































Charts	.NET Framework (WF, WPF, ASP.NET and ASP.NET MVC)	.NET Standard (UWP, .NET Core and Xamarin)
Creation and manipulation of Excel 2-D charts.		
Creation and manipulation of Excel 3-D charts.		
Creation and manipulation of custom charts.		





Creation and manipulation of sparklines.		
Creation and manipulation of Excel 2016 charts.		
Pivot Tables		
Creation and manipulation of pivot tables (while saving as XLS format, the pivot table is preserved as it is from the input XLS file).		
Apply various pivot table settings like page filter, row filter, and column filter (while saving as XLS format, pivot table is preserved as-is from the input XLS file).		
Supports pivot table settings such as visibility of field list, collapse button, field caption and field option, header captions, etc.		













Security	.NET Framework (WF, WPF, ASP.NET and ASP.NET MVC)	.NET Standard (UWP, .NET Core and Xamarin)
Open and save encrypted and decrypted documents.		
Protect and unprotect workbook.		
Protect and unprotect worksheets.		
Lock and unlock cells for write protection.		

Worksheet Features	.NET Framework (WF, WPF, ASP.NET and ASP.NET MVC)	.NET Standard (UWP, .NET Core and Xamarin)
Formatting		
Creation and manipulation of conditional formatting (supports		

icon sets, data bars, color scales, and specific date in XLSX).		
Auto fit or resize rows and columns.		
Apply or modify number formats.		
Apply or modify fill settings.		
Apply or modify font settings.		
Apply or modify cell border settings.		
Apply or modify all the cell text alignments.		
Excel 2007 themes.		
Copy/Paste		
Add or copy worksheets within or across workbooks.		
Copy a range to another range across worksheets.		
Copy a range with different copy options.		
Move a range to another range across worksheets.		
Data		
Creation and manipulation of data validation.		
Apply or modify AutoFilters to filter worksheet data.		
Apply or modify data sorting.		
Find and replace data.		
Row/Column Manipulation		
Apply or modify view settings to freeze, unfreeze, and split panes.		

Hide or unhide rows and columns.		
Merge and unmerge cells.		
Auto-Fit Rows and Columns.		
Insert		
Insertion and deletion of images in worksheets.		
Apply or modify background image in a worksheet.		
Add, remove and modify hyperlinks.		
Group or ungroup rows and columns.		
Add or modify subtotals of grouped data.		
Addition and deletion of OLE objects.		
Page Setup		
Get or set header and footer settings.		
Apply or modify horizontal and vertical page breaks.		
Apply or modify custom page breaks.		
Apply various page setup options like paper size, orientation, scaling, margins, etc.		
Apply print settings.		
Shapes		
Creation and manipulation of AutoShapes.		
Creation and manipulation of check boxes.		





Creation and manipulation of combo boxes.		
Creation and manipulation of comments.		
Creation and manipulation of text boxes.		
Miscellaneous		
Preserve Macros		




























Conversion	.NET Framework (WF, WPF, ASP.NET and ASP.NET MVC)	.NET Standard (UWP, .NET Core and Xamarin)
Converts Excel to PDF.		
Converts an entire Excel worksheet into a single image.		
Converts specific range of an Excel worksheet into a single image.		
Converts Excel chart to image.		
Converts Excel to HTML.		
Converts Excel document to ODS (Open Document Spreadsheet).		













Note: Worksheet to image conversion is supported from .NET Framework 2.0 and .NET Standard 1.4 onwards.











Supported Features in Blazor Platform

The list of supported and non-supported Excel features in Blazor platform is given below.

Features	Blazor Server-Side	Blazor Client-Side
Excel elements - text, image, table, auto filter, data sort, hyperlink, data validation, conditional formatting, formula, pivot table, comment, shape, chart, grouping, page setting and more.		
Presentation formatting - text, table, conditional formatting, styles.		

Create an Excel workbook from scratch.		
Open an existing Excel workbook and edit.		
Read worksheets on demand when Excel file contains multiple worksheets.		
Import data from DataTable, DataColumn, DataView, Array, Collection Objects and Nested Collection Objects to worksheet.		
Import data into a pre-formatted template using Template Markers.		
Export data from worksheet to datatable, nested class objects and nested collection objects.		
Access or modify built-in document properties.		
Read and write Excel formulas.		
Encryption and decryption.		
Find and replace text.		
Add or copy worksheets within or across workbooks.		
Copy a range to another range across worksheets.		
Move a range to another range across worksheets.		
Save chart as image.		

File formats	Blazor Server-Side		Blazor Client-Side	
	Open	Save	Open	Save
Excel 97-2003 formats (.XLS, .XLT)				
Excel 2007 and above formats (.XLSX, .XLTX, *.XLSM)				
XML Spreadsheet 2003 (* .XML)				

CSV				
Excel to PDF	N/A		N/A	
Excel to HTML	N/A		N/A	
Worksheet to Image (PNG, JPEG)	N/A		N/A	

Improving Performance

This section gives you an idea for improving performance while developing with XlsIO.

UsedRange

Get **UsedRange** globally. It is recommended to get the UsedRange in loops as follows

C#

```
int lastRow = sheet.UsedRange.LastRow;
for(int i=0;i<lastRow;i++)
{
    //codes
}
//Do not use like below.
for(int i = 0;i<sheet.UsedRange.LastRow;i++)
{
    //codes
}
```

VB.NET

```
Dim lastRow As Integer = sheet.UsedRange.LastRow
For i As Integer = 0 To lastRow - 1
    'codes
Next
'Do not use like below.
For i As Integer = 0 To sheet.UsedRange.LastRow - 1
    'codes
Next
```

Range Access

Use **IMigrantRange** instead of **IRange** to optimize performance while dealing with large data.

The **IMigrantRange** interface can be used to access and manipulate worksheet range. This is an optimal method of writing values with better memory performance.

The following code example illustrates how the **IMigrantRange** is accessed.

C#

```
IMigrantRange migrantRange = workbook.Worksheets[0].MigrantRange;
// Writing Data.
for (int row = 1; row <= rowCount; row++)
```

```

{
    for (int column = 1; column <= colCount; column++)
    {
        // Writing values.
        migrantRange.ResetRowColumn(row, column);
        // Setting value of this migrant range which is similar to IRange object.
        migrantRange.Value = "Syncfusion";
    }
}

```

VB.NET

```

'Writing Data.
Dim row As Integer
Dim migrantRange As IMigrantRange = workbook.Worksheets(0).MigrantRange
For row = 1 To rowCount Step row + 1
    Dim column As Integer
    For column = 1 To colCount Step column + 1
        'Writing values.
        migrantRange.ResetRowColumn(row, column)
        ' Setting value of this migrant range which is similar to IRange object.
        migrantRange.Value = "Syncfusion"
    Next
Next

```

IMigrantRange provides us a **SetValue** method in which different value for the range can be assigned. Following code snippet illustrates regarding this.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
excelEngine.Excel.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = excelEngine.Excel.Workbooks.Create();
IWorksheet sheet = workbook.Worksheets[0];
IMigrantRange migrantRange = workbook.Worksheets[0].MigrantRange;
// Writing values.
migrantRange.ResetRowColumn(1, 1);
//Setting boolean value
migrantRange.SetValue(true);
migrantRange.ResetRowColumn(1, 2);
//Setting DateTime value
migrantRange.SetValue(DateTime.Now);
migrantRange.ResetRowColumn(1, 3);
//Setting double value
migrantRange.SetValue(5.5);
migrantRange.ResetRowColumn(1, 4);
//Setting int value
migrantRange.SetValue(5);
migrantRange.ResetRowColumn(1, 5);
//Setting string value
migrantRange.SetValue("Syncfusion");
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("MigrantRange.xlsx");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
excelEngine.Excel.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Create()
Dim sheet As IWorksheet = workbook.Worksheets(0)
Dim migrantRange As IMigrantRange = workbook.Worksheets(0).MigrantRange
' Writing values.
migrantRange.ResetRowColumn(1, 1)
'Setting boolean value
migrantRange.SetValue(True)
migrantRange.ResetRowColumn(1, 2)
'Setting DateTime value
migrantRange.SetValue(DateTime.Now)
migrantRange.ResetRowColumn(1, 3)
'Setting double value
migrantRange.SetValue(5.5)
migrantRange.ResetRowColumn(1, 4)
'Setting int value
migrantRange.SetValue(5)
migrantRange.ResetRowColumn(1, 5)
'Setting string value
migrantRange.SetValue("Syncfusion")
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("MigrantRange.xlsx")
workbook.Close()
excelEngine.Dispose()

```

Styles

Use global styles, rather than using different cell styles for each cell/range. See [Applying global styles](#).

Use Begin and End call while using more than one global style for a worksheet.

C#

```

//Defining body style
IStyle bodyStyle = workbook.Styles.Add("BodyStyle");
bodyStyle.BeginUpdate();
bodyStyle.Color = Color.FromArgb(239, 243, 247);
bodyStyle.Borders[ExcelBordersIndex.EdgeLeft].LineStyle =
ExcelLineStyle.Thin;
bodyStyle.Borders[ExcelBordersIndex.EdgeRight].LineStyle =
ExcelLineStyle.Thin;
bodyStyle.EndUpdate();

```

VB.NET

```

'Defining body style
Dim bodyStyle As IStyle = workbook.Styles.Add("BodyStyle")
bodyStyle.BeginUpdate()
bodyStyle.Color = Color.FromArgb(239, 243, 247)
bodyStyle.Borders(ExcelBordersIndex.EdgeLeft).LineStyle =
ExcelLineStyle.Thin

```

```
bodyStyle.Borders(ExcelBordersIndex.EdgeRight).LineStyle =
ExcelLineStyle.Thin
bodyStyle.EndUpdate()
```

AutoFit

Minimize AutoFit manipulations which reduces the time consumption.

For improved performance in Excel to PDF conversion, it is recommended to set the **IApplication.SkipAutoFitRow** property as TRUE.

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
//Skips AutoFitting of rows during conversion
application.SkipAutoFitRow = true;
```

VB.NET

```
Dim excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
'Skips AutoFitting of rows during conversion
application.SkipAutoFitRow = True
```

Importing DataTable

ImportDataTable overload method which has **ImportOnSave** argument allows you to import data with less memory consumption along with improved method performance by serializing the data directly on save method. This option is preferred for larger data that need to be imported in short time.

C#

```
DataTable table = Worksheet.ExportDataTable(1, 1,
Worksheet.UsedRange.LastRow, Worksheet.UsedRange.LastColumn,
ExcelExportDataTableOptions.DetectColumnTypes);
//Enable ImportOnSave option along with column header.
workbook.Worksheets[0].ImportDataTable(table, 1, 1, true, true);
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs("Output.xlsx");
```

VB.NET

```
Dim table As DataTable = Worksheet.ExportDataTable(1, 1,
Worksheet.UsedRange.LastRow, Worksheet.UsedRange.LastColumn,
ExcelExportDataTableOptions.DetectColumnTypes)
'Enable ImportOnSave option along with column header.
workbook.Worksheets(0).ImportDataTable(table, 1, 1, True, True)
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs("Output.xlsx")
```

Limitations

- Cannot modify data dynamically
- Styles cannot be applied

- Table style cannot be applied
- Existing sheet data will be lost

Data Validation

Use of BeginUpdate and EndUpdate methods for large blocks of Data Validation greatly improves the performance.

C#

```
// List data validation for entire column
IDataValidation validation = sheet.Range["A3"].EntireColumn.DataValidation;
validation.BeginUpdate();
validation.DataRange = sheet.Range["D1:D56"];
validation.IsEmptyCellAllowed = true;
validation.IsListInFormula = false;
validation.EndUpdate();
```

VB.NET

```
' List data validation for entire column
Dim validation As IDataValidation =
sheet.Range("A3").EntireColumn.DataValidation
validation.BeginUpdate()
validation.DataRange = sheet.Range("D1:D56")
validation.IsEmptyCellAllowed = True
validation.IsListInFormula = False
validation.EndUpdate()
```

Known Exceptions Details

The list of known exceptions thrown in Essential XlsIO is listed below.

ApplicationException

Class	Message	Reason
Workbook	"Workbook was not created from file." + " That is why workbook file not specified." + " You must use SaveAs method instead."	The file name cannot be empty
Worksheet	"Can't deselect all worksheets."	Can't deselect all worksheets
	"Can't find data for MSODrawing"	The value of current mso index cannot greater than array drawings count
	"Sheet is already protected, before use unprotect method"	The property is true the sheet is already protected
	"You cannot use this command on a protected sheet"	The condition is false you cannot use this command on protected sheet

ArgumentException

Class	Message	Reason
Pictures	string cannot be empty.	Valid filename of the image must be provided.
Shape	From center style support only <i>var1 or var2</i>	From center style support only <i>var1 or var2</i>
	strName cannot be null or empty.	String cannot be empty
	strShapeName - string cannot be empty.	String cannot be empty
	The rotation value should be between -3600 and 3600	The rotation value should be between -3600 and 3600
	Name cannot be null or empty	Name of the UserPicture cannot be empty.
	Path cannot be null or empty	Path of the UserPicture cannot be null or empty.
	Path cannot be null or empty	Path of the UserTexture cannot be empty.
	Shape name cannot be null.	Valid name is not found for the shape.
	The specified value is out of range	TextureHorizontalScale must be >= 21475.
	Â The specified value is out of range	TextureOffsetX must be >= 169056.
	Â The specified value is out of range	TextureOffsetY must be >= 169056.
	Â The specified value is out of range	TextureVerticalScale must be >= 21475.
	This method supports only preset textured	Custom gradient texture does not support.
	TransparencyColor	The TransparencyFrom on shape should not be less than 0 and greater than 1.
TextBox	"Reference is not valid	The value of the string index 0 is equal to "=" the reference is not valid
Workbook	"Can't refer to external worksheets"	Cannot refer to external workbook

	"FileName cannot be empty."	The file name cannot be empty
	"fileName"	String cannot be empty
	"name"	String cannot be empty
	"Names array must contain at least one name."	Names array must contain at least one name
	"separator"	String cannot be empty
	"strFileName - string cannot be empty."	String cannot be empty
	"strName - string cannot be empty"	String cannot be empty
	"strSheetName"	String cannot be empty
	"Workbook is protected and password wasn't specified."	Workbook is protected and password wasn't specified
	"Contains invalid characters"	Contains invalid characters
	"FileName cannot be empty."	The file name cannot be empty
	"FileName cannot be empty."	The file name cannot be empty
Worksheet	"arrDataColumns can't be empty"	String cannot be empty
	"Cannot sets formula value in cell that doesn't contain formula"	Cannot sets formula value in cell that doesn't contain formula
	"Can't insert column"	Cannot insert column given condition is false
	"Can't insert row"	Cannot insert row given condition is true
	"cellFormat"	The array extend formats count is not less than cell format index value
	"defaultStyle"	The XF index value is not equal to minimum value of the integer
	"Each range argument should contain a single cell"	Each range argument should contain a single cell
	"FileName cannot be empty."	File name cannot be empty
	"Image Path doesn't exist"	Image path does not exist
	"Ranges do not fit each other"	The Ranges do not fit each other
	"separator"	String cannot be null or empty
	"separator"	String cannot be null or empty

	"strName - string cannot be empty."	String cannot be empty
	"Workbook at least must contains one worksheet. You cannot remove last worksheet.", "sheet"	Workbook at least must contain one worksheet. You cannot remove last worksheet
	"Workbook must contains at least one worksheet. You cannot remove last worksheet.", "sheet"	Workbook must contain at least one worksheet. You cannot remove last worksheet
	"Invalid column index."	The column index is not less than 0 and greater than 0x3fff
	"Invalid row index."	The row index is not less than 0 and greater than 0xfffff
	"separator"	String cannot be empty
	Sheet Name is InValid	Sheet Name is InValid

ArgumentNullException

Class	Message	Reason
Workbook	"Name can't be NULL."	Name can't be NULL
	"separator"	The string cannot be empty
	"styleIndexes"	The style index is not equal to null
	"password"	String cannot be empty
Worksheet	"Conditions"	The condition property is not null and CFERecords property is not equal to null
	"firstColumn"	The column index value is not equal to 0 and greater than book maximum column count
	"password"	The string cannot be empty
	"separator"	String cannot be empty
	"dataSource"	String cannot be empty

ArgumentOutOfRangeException

Class	Message	Reason
PageSetup	Parts array must have only three elements.	The header/footer string should have 3 parts.

	The string is too long. Reduce the number of characters used.	The header/footer value exceeds the characters limit.
	Zoom value must be between 10 and 400 percent.	The zoom value in page setup should be within the given limit.
Shape	BottomRowOffset	The BottomRowOffset should not be less than 0.
	Height	The Height should not be less than 0.
	iColumn1	The column should not be less than 1 and greater than maximum column count.
	iColumn2	The column should not be less than 1 and greater than maximum column count.
	iPixels	The pixels should not be less than 0 and greater than row height.
	Can't be less than zero.	The iPixels should not be less than 0.
	iRow1	The row value should not be less than 1 and greater than maximum row count.
	iRow2	The row value should not be less than 1 and greater than maximum row count.
	LeftColumnOffset	The LeftColumnOffset should not be less than 0.
	RightColumnOffset	The RightColumnOffset should not be less than 0.
	scaleHeight	The ScaleHeight should not be less than 0.
	scaleWidth	The ScaleWidth should not be less than 0.
	TopRowOffset	The TopRowOffset should not be less than 0.
	Value	The Transparency should not be less than 0 and greater than 1.

	Weight	The Weight should not be less than 0 and greater than maximum line weight value.
	Width	The Width should not be less than 0.
	iPixels can't be less than zero.	The iPixels should not be less than 0
	Gradient degree is out of range	The gradient range should not be less than -1 and greater than 1.
	Transparency	The Transparency on shape should not be less than 0 and greater than 1.
	TransparencyFrom	The TransparencyFrom value on shape should not be less than 0 and greater than 1.
	TransparencyTo	The TransparencyTo value on shape should not be less than 0 and greater than 1.
Workbook	"ActiveSheetIndex"	The sheet value is not less than 0 and greater than list object count
	"Array cannot contain more than 4 selection records"	Array cannot contain more than 4 selection records
	"DisplayedTab", "Displayed tab must be greater than zero and less than Worksheets count"	Displayed tab must be greater than zero and less than Worksheets count
	"fileName"	The file name cannot be empty
	"fullName"	String cannot be empty
	"index"	Value cannot be less than 0 and greater than Count
	"index", "Index cannot be less than 0 and larger than Palette colors array size."	Index cannot be less than 0 and larger than Palette colors array size
	"index", "Value cannot be less than 0 and greater than Count - 1."	Value cannot be less than 0 and greater than Count - 1
	"iRef"	The reference index value is not less than or equal to extern sheet reference and less than 0;

	"iReferenceIndex", "Value cannot be less than 0 and greater than arrRefs.Count - 1"	Value cannot be less than 0 and greater than arrRefs.Count - 1
	"iStartIndex"	The start index value is not less than stro0 and greater than color count
	"maxCount"	The XF index value is not less than or equal to 0
	"PasswordToOpen", "Password too long. Maximum password length is 15 characters."	Password too long. Maximum password length is 15 characters
	"referenceIndex", "Value cannot be less than 0 and greater than arrRefs.Count - 1"	Value cannot be less than 0 and greater than arrRefs.Count - 1
	"sheetsQuantity", "Quantity of worksheets must be greater than zero."	Quantity of worksheets must be greater than zero
	string.Format("index is {0}, Count is {1}", index, List.Count)	Index value is not less than 0 and not greater than equal to list count
	string.Format("index is {0}, Count is {1}", index, count)	Index value is not less than 0 and not greater than equal to list count
Worksheet	"Apostrophe can't be used as first and/or last character of the worksheet's name."	Apostrophe can't be used as first and/or last character of the worksheet's name
	"Chart index"	Index value is not less than 0 and not greater than equal to list count
	"column index"	The column index value is not equal to 0 and greater than book maximum column count
	"Column", "Column index cannot be larger then 256 or less then one"	Column index cannot be larger than 256 or less than one
	"columnIndex", "Value cannot be less than 0 and greater than 255"	Value cannot be less than 0 and greater than 255
	"count"	The count is not less than 0
	"drawingItemName"	The string name cannot be empty

	"firstColumn"	The column index value is not equal to 0 and greater than book maximum column count
	"firstRow"	The row index value is not equal to 0 and greater than book maximum rows count
	"iColumn can't be less than 1"	iColumn can't be less than 1
	"iColumnCount", "Value cannot be less 1 and greater than max column index"	Value cannot be less 1 and greater than max column index
	"iColumnIndex", "Column index is out of range."	Column index is out of range
	"iColumnIndex", "Value cannot be less 1 and greater than max column index."	Value cannot be less 1 and greater than max column index.
	"iColumnIndex", "Value cannot be less than 1 and greater than m_book.MaxColumnCount."	Value cannot be less than 1 and greater than m_book.MaxColumnCount
	"iCondFmtPos"	The index value is not less than 0
	"iCustomPropertyPos"	Index value is not less than 0 and not greater than equal to list count
	"iDValPos"	The integer value is not less than 0
	"index < 0"	The index value is not less than 0
	"index"	Index value is not less than 0 and not greater than equal to list count
	"index", "Value cannot be less than 0 and greater than Count - 1."	Value cannot be less than 0 and greater than Count - 1
	"iNewIndex"	Index value is not less than 0 and not greater than equal to list count
	"iOldIndex"	Index value is not less than 0 and not greater than equal to list count
	"iRowIndex"	The row index value is not equal to 0 and greater than book maximum rows count
	"iRowIndex", "Value cannot be less 1 and greater than max row index."	Value cannot be less 1 and greater than max row index

"iRowIndex", "Value cannot be less than 1 and greater than m_book.MaxColumnCount."	Value cannot be less than 1 and greater than m_book.MaxColumnCount
"iSTIndex"	The shared string index value is not less than 0 or greater than book inner shared string count
"iStartIndex"	The index value is not less than 0
"Length of the password can't be more than " + DEFMAXPASSWORDLEN	The password length value is greater than default maximum password length
"maxCount"	The index value is not less than or equal to 0
"range"	The range of the row and column is not equal to range of the last column and last row
"rowIndex"	The row index value is not less than 1 and greater than book maximum count
"Standard Row Height"	The standard row height value is not less than 0
"Text value cannot be null or empty"	Text value cannot be null or empty
"Text value cannot be null or empty. First symbol must be '#'"	Text value cannot be null or empty. First symbol must be '#'
"Text value cannot be null or empty. First symbol of formula cannot be '='"	Text value cannot be null or empty. First symbol of formula cannot be '='
"Value cannot be less 1 and greater than max column index."	Value cannot be less 1 and greater than maximum column index
"Value does not valid error string."	Given value does not contains dictionary
"Worksheets collection does not contain specified worksheet."	Worksheets collection does not contain specified worksheet
"Zoom", "Zoom must be in range from 10 till 400."	Zoom must be in range from 10 till 400.
strParentItemName	The string name cannot be empty

	Column value cannot be less than 0 and greater than 255	Value cannot be less than 0 and greater than 255
	End Row Index cannot be greater than max row index	Value cannot be greater than max row index
	firstColumn	The column index value is not equal to 0 and greater than book maximum column count
	firstRow	The row index value is not equal to 0 and greater than book maximum rows count
	Length of the password can't be more than password length	The password length value is greater than default maximum password length
	Row Index value cannot be less than 1 and greater than max row index	The row index value is not equal to 0 and greater than book maximum rows count
	Standard Column Width.	The standard column width value is not less than 0 and greater than default maximum column width
	Value cannot be less 1 and greater than max column index.	Value cannot be less 1 and greater than max column index
	Value cannot be less 1 and greater than max row index.	Value cannot be less 1 and greater than max row index
	Value must be 0 to 3	Value must be 0 to 3
	string.Format("index is {0}, Count is {1}", index, List.Count)	Index value is not less than 0 and not greater than equal to list count

ExcelWorkbookNotSavedException

Class	Message	Reason
Workbook	"Object cannot be disposed." + " Save workbook or set property ThrowNotSavedOnDestroy to false.	Save workbook or set property ThrowNotSavedOnDestroy to false

FileNotFoundException

Class	Message	Reason
Shape	File represents by current path doesn't exist	File represents by current path doesn't exist.

Workbook	"File could not be found. Please verify the file path."	"File could not be found."
	string.Format("File {0} could not be found. Please verify the file path.", filename)	The file name is not there in given file path the exception will throw

InvalidRangeException

Class	Message	Reason
Worksheet	Can't copy to destination range	The destination array formula not separator the cannot copy to destination range

NotSupportedException

Class	Message	Reason
Shape	This property can be set only when Gradient Style is selected.	This property can be set only when Gradient Style is selected.
	This property supports only if pattern style is checked	This property supports only if pattern style is checked.
	This property supports only if pattern style is checked	This property supports only if texture style is checked.
	This property support only if defined user texture of picture	This property support only if defined user texture of picture.
	"This property supports only if Checked Solid style."	This property supports only if Solid style is checked
	"This shape doesn't support fill properties."	The m_bSupportOption is false the fill format not supported
	"This shape doesn't support line properties."	The m_bSupportOption is false the line format not supported
	HyperLink	The shape is not equal to AutoShape, picture, textbox does not supported in hyperlink
	This property supported only if checked preset color type	Supports only Excel gradient preset.
	This property supports only if checked one color gradient	Supports only Excel gradient preset.
	This variant doesn't support center shading style.	The shape object doesn't support center shading style.

Workbook	"Name of worksheet must be unique in a workbook."	Name of worksheet must be unique in a workbook
	"Not supported encryption type."	This property is not supported for encryption type
	"Weak encryption algorithm is not supported."	Weak encryption algorithm is not supported.
Worksheet	Name of worksheet must be unique in a workbook	Name of worksheet must be unique in a workbook

FAQ Section

The frequently asked questions in Essential XlsIO are listed below.

How to open an existing XLSX workbook and save it as XLS?

You can open and save an existing .xlsx file to the .xls file by using XlsIO. The following code snippet illustrates this.

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
//Open an existing Excel 2013 file.
IWorkbook workbook = excelEngine.Excel.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
//Save it as "Excel 97 to 2003" format.
workbook.Version = ExcelVersion.Excel97to2003;
workbook.SaveAs("Output.xls");
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
'Open an existing Excel 2013 file.
Dim workbook As IWorkbook = excelEngine.Excel.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
'Save it as "Excel 97 to 2003" format.
workbook.Version = ExcelVersion.Excel97to2003
workbook.SaveAs("Output.xls")
workbook.Close()
excelEngine.Dispose()
```

Note: Workbook must be saved in appropriate version, failing in this leads to file corruption.

How to open an Excel file from Stream?

XlsIO provides support for opening a file that is stored as a stream. The following code snippet illustrates this.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Opening a File from a Stream
FileStream fileStream = new FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
IWorkbook workbook = application.Workbooks.Open(fileStream);
workbook.SaveAs("Output.xlsx");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
'Opening a File from a Stream
Dim fileStream As New FileStream("Sample.xlsx", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite)
Dim workbook As IWorkbook = application.Workbooks.Open(fileStream)
workbook.SaveAs("Output.xlsx")
workbook.Close()
excelEngine.Dispose()

```

How to save a file to stream?

XlsIO provides support to save a workbook to a .NET stream. The following code snippet illustrates this.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenTypeAutomatic);
//Save the workbook to stream.
FileStream fileStream = new FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite, FileShare.ReadWrite);
workbook.SaveAs(fileStream);
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenTypeAutomatic)
'Save the workbook to stream.
Dim fileStream As New FileStream("Output.xlsx", FileMode.Create,
FileAccess.ReadWrite, FileShare.ReadWrite)
workbook.SaveAs(fileStream)
workbook.Close()
excelEngine.Dispose()

```

How to create and open Excel Template files by using XlsIO?

Creating Excel Template Files

Excel template files (XLT or XLTX) can be created in XlsIO by saving a file as **Template** using **ExcelSaveType** enumeration. The following code snippet illustrates this.

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Save as XLT.
workbook.Version = ExcelVersion.Excel97to2003;
workbook.SaveAs("XLTFFile.xlt", ExcelSaveType.SaveAsTemplate);
//Save as XLTX.
workbook.Version = ExcelVersion.Excel2007;
workbook.SaveAs("XLTXFile.xltx", ExcelSaveType.SaveAsTemplate);
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Save as XLT.
workbook.Version = ExcelVersion.Excel97to2003
workbook.SaveAs("XLTFFile.xlt", ExcelSaveType.SaveAsTemplate)
'Save as XLTX.
workbook.Version = ExcelVersion.Excel2007
workbook.SaveAs("XLTXFile.xltx", ExcelSaveType.SaveAsTemplate)
workbook.Close()
excelEngine.Dispose()
```

Opening Excel Template Files

In XlsIO, an Excel template file is opened in the same way, as excel workbook (.xls and .xlsx) is opened. The following code snippet illustrates this.

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Open Excel Template.
IWorkbook workbook = application.Workbooks.Open("Sample.xltx",
ExcelOpenType.Automatic);
workbook.SaveAs("Output.xlsx");
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
'Open Excel Template.
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xltx",
ExcelOpenType.Automatic)
workbook.SaveAs("Output.xlsx")
workbook.Close()
excelEngine.Dispose()

```

How to open an Excel 2013 Macro Enabled Template?

You can open and save an Excel 2013 Macro Enabled Template to XLSM (Excel 2013 Macro Enabled Document) format. The following code snippet illustrates this.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Open an existing XLTM file.
IWorkbook workbook = application.Workbooks.Open("Sample.xlsm",
ExcelOpenType.Automatic);
//Save the file as XLSM.
workbook.SaveAs("Output.xlsm");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
'Open an existing XLTM file.
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsm",
ExcelOpenType.Automatic)
'Save the file as XLSM.
workbook.SaveAs("Output.xlsm")
workbook.Close()
excelEngine.Dispose()

```

How to change the grid line color of the Excel sheet?

In Essential XlsIO, you can change the grid line color of the worksheet using **GridLineColor** property. The below code snippet illustrate this.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//To change the grid line color using ExcelKnownColors
worksheet.GridLineColor = ExcelKnownColors.Blue;

```

```
workbook.SaveAs("GridLineColor.xlsx");
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'To change the grid line color using ExcelKnownColors
worksheet.GridLineColor = ExcelKnownColors.Blue
workbook.SaveAs("GridLineColor.xlsx")
workbook.Close()
excelEngine.Dispose()
```

How to copy and paste the values of the cells that contain only formulas?

You can copy and paste the values of the cell which contain only formula using CopyTo method by specifying the ExcelCopyRangeOptions as None. The following code snippet illustrates this.

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Assigning formula to a cell
worksheet.Range["A3"].Formula="SUM(1+1)";
IRange sourceRange = worksheet.Range["A3"];
IRange destinationRange = worksheet.Range["B1"];
//Copy and paste the values using ExcelCopyRangeOption
sourceRange.CopyTo(destinationRange, ExcelCopyRangeOptions.None);
workbook.SaveAs("Output.xlsx");
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Assigning formula to a cell
worksheet.Range("A3").Formula = "SUM(1+1)"
Dim sourceRange As IRange = worksheet.Range("A3")
Dim destinationRange As IRange = worksheet.Range("B1")
'Copy and paste the values using ExcelCopyRangeOption
sourceRange.CopyTo(destinationRange, ExcelCopyRangeOptions.None)
workbook.SaveAs("Output.xlsx")
workbook.Close()
excelEngine.Dispose()
```

How to copy a range from one workbook to another?

You can copy the range from source workbook to the destination workbook through CopyTo method.

The following code snippet illustrates this.

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook sourceWorkbook = application.Workbooks.Open("SourceWorkbook.xlsx",
ExcelOpenType.Automatic);
IWorkbook destinationWorkbook =
application.Workbooks.Open("DestinationWorkbook.xlsx",
ExcelOpenType.Automatic);
IWorksheet SourceWorksheet = SourceWorkbook.Worksheets[0];
//The first worksheet object in the worksheets collection in the Destination
Workbook is accessed.
IWorksheet DestinationWorksheet = DestinationWorkbook.Worksheets[0];
//Assigning an object to the range of cells (90 rows) both for source and
destination.
IRange sourceRange = SourceWorksheet.Range[1, 1, 90, 100];
IRange destinationRange = DestinationWorksheet.Range[1, 1, 90, 100];
//Copying (90 rows) from Source to Destination worksheet.
sourceRange.CopyTo(destinationRange);
destinationWorkbook.SaveAs("CopyingRange.xlsx");
destinationWorkbook.Close();
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As ExcelEngine = New ExcelEngine
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim sourceWorkbook As
IWorkbook = application.Workbooks.Open("SourceWorkbook.xlsx",
ExcelOpenType.Automatic)
Dim destinationWorkbook As IWorkbook =
application.Workbooks.Open("DestinationWorkbook.xlsx",
ExcelOpenType.Automatic)
'The first worksheet object in the worksheets collection in the Source
Workbook is accessed.
Dim SourceWorksheet As Syncfusion.XlsIO.IWorksheet =
SourceWorkbook.Worksheets(0)
'The first worksheet object in the worksheets collection in the Destination
Workbook is accessed.
Dim DestinationWorksheet As Syncfusion.XlsIO.IWorksheet =
DestinationWorkbook.Worksheets(0)
'Assigning an object to the range of cells (90 rows) both for source and
destination.
Dim sourceRange As Syncfusion.XlsIO.IRange = SourceWorksheet.Range(1, 1, 90,
100)
Dim destinationRange As Syncfusion.XlsIO.IRange =
DestinationWorksheet.Range(1, 1, 90, 100)
'Copying (90 rows) from Source to Destination worksheet.
sourceRange.CopyTo(destinationRange)
destinationWorkbook.SaveAs("CopyingRange.xlsx")
destinationWorkbook.Close()
```

```
excelEngine.Dispose()
```

How to merge several excel files from more than one workbook to a single file?

You can merge several excel files from more than one work book to a single file. The following code snippet illustrates this.

C#

```
//Loads the all template document from Data folder.
string[] files = Directory.GetFiles(@"../../Data/");
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
//Create empty Excel workbook instance with one empty worksheet
IWorkbook workbook = application.Workbooks.Create(1);
//Enumerates all the workbook files from the data folder and clone and merge
it into new workbook.
foreach (string file in files)
{
    //Loads the all template document from data folder.
    FileStream inputStream = new FileStream(file, FileMode.Open,
    FileAccess.Read);
    //Opens the template workbook from stream
    IWorkbook tempWorkbook = application.Workbooks.Open(inputStream);
    //Disposes the stream
    inputStream.Dispose();
    //Cloning all workbook's worksheets
    workbook.Worksheets.AddCopy(tempWorkbook.Worksheets);
}
//removing the first empty worksheet
workbook.Worksheets.Remove(0);
workbook.SaveAs("MergingFiles.xlsx");
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
'Loads the all template document from data folder.
Dim files As String() = Directory.GetFiles("../../Data/")
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
'Create empty Excel workbook instance with one empty worksheet
Dim workbook As IWorkbook = application.Workbooks.Create(1)
'Enumerates all the workbook files from the data folder and clone and merge
it into new workbook.
For Each file As String In files
    'Loads the all template document from data folder.
    Dim inputStream As New FileStream(file, FileMode.Open, FileAccess.Read)
    'Opens the template workbook from stream
    Dim tempWorkbook As IWorkbook = application.Workbooks.Open(inputStream)
    'Disposes the stream
    inputStream.Dispose()
    'Cloning all workbook's worksheets
    workbook.Worksheets.AddCopy(tempWorkbook.Worksheets)
```

Next

```
'removing the first empty worksheet
workbook.Worksheets.Remove(0)
workbook.SaveAs("MergingFiles.xlsx")
workbook.Close()
excelEngine.Dispose()
```

How to ignore the green error marker in worksheets?

When there exists data that are of different formats, the error marker appears in cells. In XlsIO You can ignore this by using **IgnoreErrorOptions** property. The following code snippet illustrate this.

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Ignore Error Options.
worksheet.Range["B3"].IgnoreErrorOptions = ExcelIgnoreError.All;
workbook.SaveAs("IgnoreGreenError.xlsx");
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Ignore Error Options.
worksheet.Range("B3").IgnoreErrorOptions = ExcelIgnoreError.All
workbook.SaveAs("IgnoreGreenError.xlsx")
workbook.Close()
excelEngine.Dispose()
```

How to protect certain cells in a worksheet?

All the cells in an Excel worksheet have a **Locked** property, which determines if the cell will be editable. When a worksheet is protected, all the cells in the worksheet get locked, by default.

However, there is often a need to protect only certain cells in a worksheet. In this scenario, you need to protect a worksheet, and set the **IsLocked** property as false for the cells that need to be made editable.

The following code snippet illustrate this.

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
```



```

IWorksheet worksheet = workbook.Worksheets[0];
//Sample data
worksheet.Range["A1:K20"].Text = "Locked";
//A1:A10 will not be protected, hence it is editable.
worksheet.Range["A1:A10"].CellStyle.Locked = false;
worksheet.Range["A1:A10"].Text = "UnLocked";
worksheet.Protect("syncfusion", ExcelSheetProtection.All);
workbook.SaveAs("ProtectCells.xlsx");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Sample data
worksheet.Range("A1:K20").Text = "Locked"
'A1:A10 will not be protected.
worksheet.Range("A1:A10").CellStyle.Locked = False
worksheet.Range("A1:A10").Text = "UnLocked"
worksheet.Protect("syncfusion", ExcelSheetProtection.All)
workbook.SaveAs("ProtectCells.xlsx")
workbook.Close()
excelEngine.Dispose()

```

Note: Locking/Unlocking cells in an unprotected worksheet has no effect.

How to set a line break inside a cell?

In order to set a line break inside a cell, you have to enable Text Wrapping for the cell, and then break the text. The following code snippet illustrates this.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Save the line break inside the cell
worksheet.Range["A1"].CellStyle.WrapText = true;
worksheet.Range["A1"].Text = String.Format("Hello\nworld");
workbook.SaveAs("LineBreak.xlsx");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)

```

```

Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Save the line break inside the cell
worksheet.Range("A1").CellStyle.WrapText = True
worksheet.Range("A1").Text = String.Format("Hello" & vbLf & "world")
workbook.SaveAs("LineBreak.xlsx")
workbook.Close()
excelEngine.Dispose()

```

How to set or format a Header/Footer?

Script commands are used to set header/ footer formatting. The following code snippet illustrate this. For more information on formatting the string, see [Inserting and Formatting Text in Headers and Footers](#)

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Format the header
worksheet.PageSetup.CenterHeader = @"&"Gothic,bold"&"Center Header Text";
workbook.SaveAs("HeaderFormat.xlsx");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Format the header
worksheet.PageSetup.CenterHeader = "&"Gothic,bold"&"Center Header Text"
workbook.SaveAs("HeaderFormat.xlsx")
workbook.Close()
excelEngine.Dispose()

```

Note: Go to “View -> Page Layout” option to view the header and footer in Microsoft Excel.

How to set print titles?

Printing Title Rows

XlsIO allows to designate row header to repeat on all pages of a printed workbook using **PrintTitleRows** property. The following code snippet illustrates this.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Print Rows 1 to 3 on every printed page.

```

```
worksheet.PageSetup.PrintTitleRows = "$A$1:$IV$3";
workbook.SaveAs ("TitleRows.xlsx");
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Print Rows 1 to 3 on every printed page.
worksheet.PageSetup.PrintTitleRows = "$A$1:$IV$3"
workbook.SaveAs ("TitleRows.xlsx")
workbook.Close()
excelEngine.Dispose()
```

Printing Title Columns

XlsIO allows to designate column header to repeat on all pages of a printed workbook using **PrintTitleColumns** property. The following code illustrates printing Title Columns.

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Print Columns 1 to 3 on every printed page.
worksheet.PageSetup.PrintTitleColumns = "$A$1:$C$65536";
workbook.SaveAs ("TitleColumns.xlsx");
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Print Columns 1 to 3 on every printed page.
worksheet.PageSetup.PrintTitleColumns = "$A$1:$C$65536"
workbook.SaveAs ("TitleColumns.xlsx")
workbook.Close()
excelEngine.Dispose()
```

For information on Print settings, refer to section [Page Setup Settings](#).

How to unfreeze the rows and columns in XlsIO?

You can unfreeze rows and columns in XlsIO by using the RemovePanes method. The following code snippet illustrates this.

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Freeze the panes.
worksheet.Range[8, 1].FreezePanes();
//Unfreeze the panes
worksheet.RemovePanes();
workbook.SaveAs("Unfreeze.xlsx");
workbook.Close();
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Freeze the panes.
worksheet.Range(8, 1).FreezePanes()
'Unfreeze the panes
worksheet.RemovePanes()
workbook.SaveAs("Unfreeze.xlsx")
workbook.Close()
excelEngine.Dispose()
```

What is the maximum range of Rows and Columns?

XlsIO has support below worksheet size for Excel 97 to 2003, Excel 2007 and later versions.

- **Excel 97 to 2003(.xls format)** – 65,536 by 256 rows and columns.
- **Excel 2007 and Later versions(.xlsx format)** – 1,048,576 by 16,384 rows and columns

The above specification is the worksheet size of Excel. For more information, see [Excel specifications and limits](#)

How to use Named Ranges with XlsIO?

A named range can be added to worksheet or workbook based on the required scope, the following code snippet illustrate this. For more information, see [Named Range](#)

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
```

```

IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Adding named range to the workbook
IName workbookName = workbook.Names.Add("WorkBookName");
workbookName.RefersToRange = worksheet.Range["I8"];
//Looping through the Named Ranges in a workbook.
foreach (IName workbookName in workbook.Names)
{
    MessageBox.Show(workbookName.Name.ToString());
}
//Adding named range to the worksheet
IName worksheetName = worksheet.Names.Add("WorkSheetName");
worksheetName.RefersToRange = worksheet.Range["J8"];
//Looping through the Named Ranges in a worksheet.
foreach (IName name in worksheet.Names)
{
    MessageBox.Show(name.Name.ToString());
}
workbook.SaveAs("NamedRange.Xlsx");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Adding named range to the workbook
Dim workbookName__1 As IName = workbook.Names.Add("WorkBookName")
workbookName__1.RefersToRange = worksheet.Range("I8")
'Looping through the Named Ranges in a workbook.
For Each workbookName__2 As IName In workbook.Names
    MessageBox.Show(workbookName__2.Name.ToString())
Next
'Adding named range to the worksheet
Dim worksheetName__3 As IName = worksheet.Names.Add("WorkSheetName")
worksheetName__3.RefersToRange = worksheet.Range("J8")
'Looping through the Named Ranges in a worksheet.
For Each worksheetName__4 As IName In worksheet.Names
    MessageBox.Show(worksheetName__4.Name.ToString())
Next
workbook.SaveAs("NamedRange.Xlsx")
workbook.Close()
excelEngine.Dispose()

```

How to add chart labels to scatter points?

The following code illustrates adding chart labels to the scatter points of the chart.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;

```

```

IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Get the chart from the charts collection
IChart chart = worksheet.Charts[0];
//Get the first series from the Series collection
IChartSerie serieOne = chart.Series[0];
//Set the Series name to the Data Labels through Data Points
serieOne.DataPoints[0].DataLabels.IsSeriesName = true;
//Set the Value to the Data Labels through Data Points
serieOne.DataPoints[0].DataLabels.IsValue = true;
workbook.SaveAs("ChartLabels.xlsx");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Get the chart from the charts collection
Dim chart As IChart = worksheet.Charts(0)
'Get the first series from the Series collection
Dim serieOne As IChartSerie = chart.Series(0)
'Set the Series name to the Data Labels through Data Points
serieOne.DataPoints(0).DataLabels.IsSeriesName = True
'Set the Value to the Data Labels through Data Points
serieOne.DataPoints(0).DataLabels.IsValue = True
workbook.SaveAs("ChartLabels.xlsx")
workbook.Close()
excelEngine.Dispose()

```

How to create a Chart with a discontinuous range?

The following code example illustrates creating a chart with discontinuous data ranges.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Entering the data for the chart.
worksheet.Range["A1"].Text = "Texas books Unit sales";
worksheet.Range["A1:D1"].Merge();
worksheet.Range["A1"].CellStyle.Font.Bold = true;
worksheet.Range["B2"].Text = "Jan";
worksheet.Range["C2"].Text = "Feb";
worksheet.Range["D2"].Text = "Mar";
worksheet.Range["A3"].Text = "Austin";
worksheet.Range["A4"].Text = "Dallas";
worksheet.Range["A5"].Text = "Houston";
worksheet.Range["A6"].Text = "San Antonio";

```

```

worksheet.Range["B3"].Number = 53.75;
worksheet.Range["B4"].Number = 52.85;
worksheet.Range["B5"].Number = 59.77;
worksheet.Range["B6"].Number = 96.15;
worksheet.Range["C3"].Number = 79.79;
worksheet.Range["C4"].Number = 59.22;
worksheet.Range["C5"].Number = 10.09;
worksheet.Range["C6"].Number = 73.02;
worksheet.Range["D3"].Number = 26.72;
worksheet.Range["D4"].Number = 33.71;
worksheet.Range["D5"].Number = 45.81;
worksheet.Range["D6"].Number = 12.17;
worksheet.Range["F1"].Number = 26.72;
worksheet.Range["F2"].Number = 33.71;
worksheet.Range["F3"].Number = 45.81;
worksheet.Range["F4"].Number = 12.17;
//Discontinuous range.
IRanges rangesOne = worksheet.CreateRangesCollection();
rangesOne.Add(worksheet.Range["B3:B6"]);
rangesOne.Add(worksheet.Range["F1:F2"]);
IRanges rangesTwo = worksheet.CreateRangesCollection();
rangesTwo.Add(worksheet.Range["D3:D6"]);
rangesTwo.Add(worksheet.Range["F3:F4"]);
//Adding a New (Embedded chart) to the Worksheet.
IChartShape shape = worksheet.Charts.Add();
shape.PrimaryCategoryAxis.Title = "City";
shape.PrimaryValueAxis.Title = "Sales (in Dollars)";
shape.ChartTitle = "Texas Books Unit Sales";
//Setting the Series Names in a Legend.
IChartSerie serieOne = shape.Series.Add();
serieOne.Name = "Jan";
serieOne.Values = rangesOne;
IChartSerie serieTwo = shape.Series.Add();
serieTwo.Name = "March";
serieTwo.Values = rangesTwo;
//Setting the (Rows & Columns) Property for the Embedded chart.
shape.BottomRow = 40;
shape.TopRow = 10;
shape.LeftColumn = 3;
shape.RightColumn = 15;
workbook.SaveAs("DiscontinuousRange.xlsx");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Entering the data for the chart.
worksheet.Range("A1").Text = "Texas books Unit sales"
worksheet.Range("A1:D1").Merge()
worksheet.Range("A1").CellStyle.Font.Bold = True
worksheet.Range("B2").Text = "Jan"

```

```

worksheet.Range("C2").Text = "Feb"
worksheet.Range("D2").Text = "Mar"
worksheet.Range("A3").Text = "Austin"
worksheet.Range("A4").Text = "Dallas"
worksheet.Range("A5").Text = "Houston"
worksheet.Range("A6").Text = "San Antonio"
worksheet.Range("B3").Number = 53.75
worksheet.Range("B4").Number = 52.85
worksheet.Range("B5").Number = 59.77
worksheet.Range("B6").Number = 96.15
worksheet.Range("C3").Number = 79.79
worksheet.Range("C4").Number = 59.22
worksheet.Range("C5").Number = 10.09
worksheet.Range("C6").Number = 73.02
worksheet.Range("D3").Number = 26.72
worksheet.Range("D4").Number = 33.71
worksheet.Range("D5").Number = 45.81
worksheet.Range("D6").Number = 12.17
worksheet.Range("F1").Number = 26.72
worksheet.Range("F2").Number = 33.71
worksheet.Range("F3").Number = 45.81
worksheet.Range("F4").Number = 12.17
'Discontinuous range.
Dim rangesOne As IRanges = worksheet.CreateRangesCollection()
rangesOne.Add(worksheet.Range("B3:B6"))
rangesOne.Add(worksheet.Range("F1:F2"))
Dim rangesTwo As IRanges = worksheet.CreateRangesCollection()
rangesTwo.Add(worksheet.Range("D3:D6"))
rangesTwo.Add(worksheet.Range("F3:F4"))
'Adding a New (Embedded chart) to the Worksheet.
Dim shape As IChartShape = worksheet.Charts.Add()
shape.PrimaryCategoryAxis.Title = "City"
shape.PrimaryValueAxis.Title = "Sales (in Dollars)"
shape.ChartTitle = "Texas Books Unit Sales"
'Setting the Series Names in a Legend.
Dim serieOne As IChartSerie = shape.Series.Add()
serieOne.Name = "Jan"
serieOne.Values = rangesOne
Dim serieTwo As IChartSerie = shape.Series.Add()
serieTwo.Name = "March"
serieTwo.Values = rangesTwo
'Setting the (Rows & Columns) Property for the Embedded chart.
shape.BottomRow = 40
shape.TopRow = 10
shape.LeftColumn = 3
shape.RightColumn = 15
workbook.SaveAs("DiscontinuousRange.xlsx")
workbook.Close()
excelEngine.Dispose()

```

How to define discontinuous ranges?

You can define a discontinuous range by adding different ranges to the Range collection. The following code example illustrates this.

C#


```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Create Range collection.
IRanges rangeCollection = worksheet.CreateRangesCollection();
//Add different ranges to the Range collection.
rangeCollection.Add(worksheet.Range["D2:D3"]);
rangeCollection.Add(worksheet.Range["D10:D11"]);
rangeCollection.Text = "Welcome";
workbook.SaveAs("DiscontinuousRange.xlsx");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Create Range collection.
Dim range As IRanges = worksheet.CreateRangesCollection()
'Add different ranges to the Range collection.
range.Add(worksheet.Range("D2:D3"))
range.Add(worksheet.Range("D10:D11"))
range.Text = "Welcome"
workbook.SaveAs("DiscontinuousRange.xlsx")
workbook.Close()
excelEngine.Dispose()

```

How to format text within a cell?

In Essential XlsIO, You can use the rich text formatting option to format the text within a cell. The following code snippet illustrates this.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Create(1);
IWorksheet worksheet = workbook.Worksheets[0];
//Insert Rich Text.
IRange range = worksheet.Range["A1"];
range.Text = "RichText";
IRichTextString richText = range.RichText;
//Formatting first 4 characters.
IFont redFont = workbook.CreateFont();
redFont.Bold = true;
redFont.Italic = true;
redFont.RGBColor = Color.Red;
richText.SetFont(0, 3, redFont);
//Formatting last 4 characters.
IFont blueFont = workbook.CreateFont();

```

```

blueFont.Bold = true;
blueFont.Italic = true;
blueFont.RGBColor = Color.Blue;
richText.SetFont(4, 7, blueFont);
workbook.SaveAs("FormattingText.xlsx");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Insert Rich Text.
Dim range As IRange = worksheet.Range("A1")
range.Text = "RichText"
Dim richText As IRichTextString = range.RichText
'Formatting first 4 characters.
Dim redFont As IFont = workbook.CreateFont()
redFont.Bold = True
redFont.Italic = True
redFont.RGBColor = Color.Red
richText.SetFont(0, 3, redFont)
'Formatting last 4 characters.
Dim blueFont As IFont = workbook.CreateFont()
blueFont.Bold = True
blueFont.Italic = True
blueFont.RGBColor = Color.Blue
richText.SetFont(4, 7, blueFont)
workbook.SaveAs("FormattingText.xlsx")
workbook.Close()
excelEngine.Dispose()

```

How to hide the summary rows and columns using XlsIO?

You can hide the summary rows and columns by using the **IsSummaryRowBelow** and **IsSummaryColumnRight** properties. The following code snippet illustrates this.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2013;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Hide the summary rows at the bottom.
worksheet.PageSetup.IsSummaryRowBelow = false;
//Hide the summary columns to the right.
worksheet.PageSetup.IsSummaryColumnRight = false;
workbook.SaveAs("SuppressRowsColumns.xlsx");
workbook.Close();
excelEngine.Dispose();

```

VB.NET

```

Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim worksheet As IWorksheet = workbook.Worksheets(0)
'Hide the summary rows at the bottom.
worksheet.PageSetup.IsSummaryRowBelow = False
'Suppress the summary columns to the right.
worksheet.PageSetup.IsSummaryColumnRight = False
workbook.SaveAs("SuppressRowsColumns.xlsx")
workbook.Close()
excelEngine.Dispose()

```

How to zip files using the Syncfusion.Compression.Zip namespace?

You can compress the file using Syncfusion.Compression.Zip namespace. The following code illustrate this.

C#

```

using Syncfusion.Compression.Zip;
ZipArchive zipArchive = new Syncfusion.Compression.Zip.ZipArchive();
zipArchive.DefaultCompressionLevel =
Syncfusion.Compression.CompressionLevel.Best;
//Add the file you want to zip.
zipArchive.AddFile("SampleFile.cs");
//Zip file name and location.
zipArchive.Save("SyncfusionCompressFileSample.zip");
zipArchive.Close();

```

VB.NET

```

Imports Syncfusion.Compression.Zip
Dim zipArchive As ZipArchive = New Syncfusion.Compression.Zip.ZipArchive()
zipArchive.DefaultCompressionLevel =
Syncfusion.Compression.CompressionLevel.Best
'Add the file you want to zip.
zipArchive.AddFile("SampleFile.cs")
'Zip file name and location.
zipArchive.Save("SyncfusionCompressFileSample.zip")
zipArchive.Close()

```

Tips: You can use CompressionLevel to reduce the size of the file.

For compressing directories, you can make use of the **AddDirectory** method which adds an empty directory file to a ZipArchive. If you want to add all the files inside the directory, then you should manually add these files by using the **AddItem** method.

The following code snippet illustrate how to add the file from the local drive.

C#

```

string fileName = @"SampleFile.cs";
ZipArchive zipArchive = new Syncfusion.Compression.Zip.ZipArchive();

```

```
zipArchive.DefaultCompressionLevel = CompressionLevel.Best;
Stream stream = new FileStream(fileName, FileMode.Open, FileAccess.Read);
FileAttributes attributes = File.GetAttributes(fileName);
ZipArchiveItem item = new ZipArchiveItem(zipArchive, "SampleFile.cs",
stream, true, attributes);
zipArchive.AddItem(item);
zipArchive.Save(@"SyncfusionCompressFileSample.zip");
zipArchive.Close();
```

VB.NET

```
Dim fileName As String = "SampleFile.cs"
Dim zipArchive As ZipArchive = New Syncfusion.Compression.Zip.ZipArchive()
zipArchive.DefaultCompressionLevel = CompressionLevel.Best
Dim stream As Stream = New FileStream(fileName, FileMode.Open,
FileAccess.Read)
Dim attributes As FileAttributes = File.GetAttributes(fileName)
Dim item As New ZipArchiveItem(zipArchive, "SampleFile.cs", stream, True,
attributes)
zipArchive.AddItem(item)
zipArchive.Save("SyncfusionCompressFileSample.zip")
zipArchive.Close()
```

How to zip all the files in subfolders using the Syncfusion.Compression.Zip namespace?

You can compress and decompress the files with our Compression library. The following code snippet illustrates this.

C#

```
using Syncfusion.Compression.Zip;
class Program
{
    private static List<DirectoryInfo> arrOfItems = new List<DirectoryInfo>();
    private static ZipArchive zipArchive = new ZipArchive();
    private static string folderPath = @"..\..\ZipFiles";
    private static void SubFoldersFiles(string path)
    {
        DirectoryInfo dInfo = new DirectoryInfo(path);
        foreach (DirectoryInfo d in dInfo.GetDirectories())
        {
            SubFoldersFiles(d.FullName);
            arrOfItems.Add(d);
        }
    }
    // Zip and save the file.
    private static void ZipAndSave()
    {
        SubFoldersFiles(folderPath);
        if (Directory.Exists(folderPath))
        {
            AddRootFiles();
            AddSubFoldersFiles();
            // Saving zipped file.
            zipArchive.Save(@"..\..\UnzippedFile.zip");
            zipArchive.Close();
        }
    }
}
```

```

Console.WriteLine("Files Zipped successfully!");
}
}
private static void AddRootFiles()
{
    string fileName = "";
    foreach (string rootFiles in Directory.GetFiles(folderPath))
    {
        //Creating the stream from file
        FileStream stream = new FileStream(rootFiles, FileMode.Open,
        FileAccess.ReadWrite);
        //Getting the File Name alone and ignoring the directory path
        fileName = Path.GetFileName(rootFiles);
        FileAttributes attribute = File.GetAttributes(rootFiles);
        zipArchive.AddItem(fileName, stream, false, attribute);
    }
}
private static void AddSubFoldersFiles()
{
    foreach (DirectoryInfo dInfo in arrOfItems)
    {
        FileInfo[] fInfo = dInfo.GetFiles();
        string mainDirectoryPath = Path.GetFullPath(folderPath);
        foreach (FileInfo file in fInfo)
        {
            //Get the File name with its current folder and ignoring the Main Directory
            string fileName = file.FullName.Replace(mainDirectoryPath, "");
            //Read the file stream by its Full name
            FileStream stream = new FileStream(file.FullName, FileMode.Open,
            FileAccess.ReadWrite);
            FileAttributes attributes = File.GetAttributes(file.FullName);
            //Add the item to the zip Archive
            zipArchive.AddItem(fileName, stream, true, attributes);
        }
    }
}
//Unzipping the Folder
private static void UnZipFiles()
{
    ZipArchive zip = new ZipArchive();
    string path = @"..\..\UnZippedFile";
    zip.Open(@"..\..\UnZippedFile.zip");
    if (!Directory.Exists(path))
    Directory.CreateDirectory(path);
    //Saving the contents of zip file to disk.
    for (int i = 0; i < zip.Count; i++)
    {
        ZipArchiveItem item = zip[i];
        string itemName = path + item.ItemName;
        //checking whether the item is root file
        if (itemName.Contains("/"))
        {
            itemName = itemName.Replace("/", "\\");
        }
        //Check whether the Directory is present or not
        if (!Directory.Exists(itemName) || itemName.Contains("\\\\"))
        {

```

```

int index = itemName.LastIndexOf("\\");
string directoryPath = itemName.Remove(index, itemName.Length - index);
Directory.CreateDirectory(directoryPath);
}
FileStream fileStream = new FileStream(itemName, FileMode.OpenOrCreate,
FileAccess.ReadWrite);
MemoryStream memoryStream = item.DataStream as MemoryStream;
memoryStream.WriteTo(fileStream);
fileStream.Flush();
fileStream.Close();
}
Console.WriteLine("File has been Unzipped");
}
static void Main(string[] args)
{
ZipAndSave();
UnZipFiles();
}
}

```

VB.NET

```

Imports Syncfusion.Compression.Zip
Class Program
Private Shared arrOfItems As New List(Of DirectoryInfo) ()
Private Shared zipArchive As New ZipArchive()
Private Shared folderPath As String = "..\..\ZipFiles"
Private Shared Sub SubFoldersFiles(path As String)
Dim dInfo As New DirectoryInfo(path)
For Each d As DirectoryInfo In dInfo.GetDirectories()
SubFoldersFiles(d.FullName)
arrOfItems.Add(d)
Next
End Sub
' Zip and save the file.
Private Shared Sub ZipAndSave()
SubFoldersFiles(folderPath)
If Directory.Exists(folderPath) Then
AddRootFiles()
AddSubFoldersFiles()
' Saving zipped file.
zipArchive.Save("..\\..\UnzippedFile.zip")
zipArchive.Close()
Console.WriteLine("Files Zipped successfully!")
End If
End Sub
Private Shared Sub AddRootFiles()
Dim fileName As String = ""
For Each rootFiles As String In Directory.GetFiles(folderPath)
'Creating the stream from file
Dim stream As New FileStream(rootFiles, FileMode.Open, FileAccess.ReadWrite)
'Getting the File Name alone and ignoring the directory path
fileName = Path.GetFileName(rootFiles)
Dim attribute As FileAttributes = File.GetAttributes(rootFiles)
zipArchive.AddItem(fileName, stream, False, attribute)
Next

```

```

End Sub
Private Shared Sub AddSubFoldersFiles()
For Each dInfo As DirectoryInfo In arrOfItems
Dim fInfo As FileInfo() = dInfo.GetFiles()
Dim mainDirectoryPath As String = Path.GetFullPath(folderPath)
For Each file__1 As FileInfo In fInfo
'Get the File name with its current folder and ignoring the Main Directory
Dim fileName As String = file__1.FullName.Replace(mainDirectoryPath, "")
'Read the file stream by its Full name
Dim stream As New FileStream(file__1.FullName, FileMode.Open,
FileAccess.ReadWrite)
Dim attributes As FileAttributes = File.GetAttributes(file__1.FullName)
'Add the item to the zip Archive
zipArchive.AddItem(fileName, stream, True, attributes)
Next
Next
End Sub
'Unzipping the Folder
Private Shared Sub UnZipFiles()
Dim zip As New ZipArchive()
Dim path As String = "..\..\UnZippedFile"
zip.Open("..\..\UnZippedFile.zip")
If Not Directory.Exists(path) Then
Directory.CreateDirectory(path)
End If
'Saving the contents of zip file to disk.
For i As Integer = 0 To zip.Count - 1
Dim item As ZipArchiveItem = zip(i)
Dim itemName As String = path + item.ItemName
'checking whether the item is root file
If itemName.Contains("/") Then
itemName = itemName.Replace("/", "\")
End If
'Check whether the Directory is present or not
If Not Directory.Exists(itemName) OrElse itemName.Contains("\") Then
Dim index As Integer = itemName.LastIndexOf("\")
Dim directoryPath As String = itemName.Remove(index, itemName.Length -
index)
Directory.CreateDirectory(directoryPath)
End If
Dim fileStream As New FileStream(itemName, FileMode.OpenOrCreate,
FileAccess.ReadWrite)
Dim memoryStream As MemoryStream = TryCast(item.DataStream, MemoryStream)
memoryStream.WriteTo(fileStream)
fileStream.Flush()
fileStream.Close()
Next
Console.WriteLine("File has been Unzipped")
End Sub
Private Shared Sub Main(args As String())
ZipAndSave()
UnZipFiles()
End Sub
End Class

```

How to protect the zip files with password using Syncfusion.Compression.Base?

Password is used for protecting files which needs more security. This can be achieved by using various encryption algorithms. The compressed zip files can be protected using encryption algorithms with password as a key for that algorithm.

Syncfusion.Compression.Base supports AES-128 bits, AES-192 bits, AES-256 bits and ZipCrypto encryption algorithms. These encryption algorithms are available under the enumeration **EncryptionAlgorithm**.

The following complete code snippet explains how to protect a zip file with password using AES-256 bits encryption algorithm.

C#

```
using Syncfusion.Compression.Zip;
class Program
{
    static void Main(string[] args)
    {
        //Initialize ZipArchive
        ZipArchive zipArchive = new ZipArchive();
        //Add files into ZipArchive
        zipArchive.AddFile("../Data/Template1.txt");
        zipArchive.AddFile("../Data/Template2.txt");
        zipArchive.AddFile("../Data/Template3.txt");
        //Protect the ZipArchive with password
        zipArchive.Protect("password", EncryptionAlgorithm.AES256);
        //Save the ZipArchive
        zipArchive.Save("WithPassword256Bit.zip");
    }
}
```

VB.NET

```
Imports Syncfusion.Compression.Zip
Module Module1
    Sub Main()
        'Initialize ZipArchive
        Dim zipArchive As ZipArchive = New ZipArchive
        'Add files into ZipArchive
        zipArchive.AddFile("../Data/Template1.txt")
        zipArchive.AddFile("../Data/Template2.txt")
        zipArchive.AddFile("../Data/Template3.txt")
        'Protect the ZipArchive with password
        zipArchive.Protect("password", EncryptionAlgorithm.AES256)
        'Save the ZipArchive
        zipArchive.Save("WithPassword256Bit.zip")
    End Sub
End Module
```

How to un-protect the zip files using Syncfusion.Compression.Base?

The following complete code snippet explains how to unprotect the zip file.

C#


```

using Syncfusion.Compression.Zip;
using System.IO;
class Program
{
    static void Main(string[] args)
    {
        //Initailize ZipArchive
        ZipArchive zipArchive = new ZipArchive();
        //Load the zip file into ZipArchive
        zipArchive.Open(new FileStream("../Data/Protected.zip", FileMode.Open),
            false, "password");
        //Unprotect the ZipArchive
        zipArchive.UnProtect();
        //Save the ZipArchive
        zipArchive.Save("WithOutPassword.zip");
    }
}

```

VB.NET

```

Imports Syncfusion.Compression.Zip
Imports System.IO
Module Module1
    Sub Main()
        'Initailize ZipArchive
        Dim zipArchive As ZipArchive = New ZipArchive
        'Load the zip file into ZipArchive
        zipArchive.Open(New FileStream("../Data/Protected.zip", FileMode.Open),
            False, "password")
        'Unprotect the ZipArchive
        zipArchive.UnProtect()
        'Save the ZipArchive
        zipArchive.Save("WithOutPassword.zip")
    End Sub
End Module

```

Does Essential XlsIO provide support for Client Profile?

Yes, Essential XlsIO provides support for Client Profile. In order to use Essential XlsIO in an application (which targeted to Client Profile), the user should include the following assemblies.

- Syncfusion.Core.dll
- Syncfusion.Compression.Base.dll
- Syncfusion.XlsIO.ClientProfile.dll

Note: Syncfusion.Core is no longer needed from 13.2 version onwards.

How to resolve the “File does not contain workbook stream” error in Syncfusion.XlsIO.Base.dll?

XlsIO does not support files generated prior to 97-2003 version. Hence the exception "File does not contain workbook stream" occurs. This can be checked in prior with the below code snippet.

C#

```

ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;

```

```
//To check whether the file is supported
var isSupported = application.IsSupported("Sample.xls");
excelEngine.Dispose();
```

VB.NET

```
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
'To check whether the file is supported
Dim isSupported = application.IsSupported("Sample.xls")
excelEngine.Dispose()
```

Note: This method is available from 12.4 version onwards.

How to resolve "Excel cannot open the file 'filename.xlsx' because the file format for the file extension is not valid. Verify that the file has not been corrupted and that the file extension matches the format of the file" error?

This error occurs when there is a mismatch between the file format and its extension. The default workbook creation version in XlsIO is Excel97-2003 (.xls). The application version set to the required version should match its file format during save, as in the below code.

C#

```
ExcelEngine excelEngine = new ExcelEngine();
IApplication application = excelEngine.Excel;
//Set application version
application.DefaultVersion = ExcelVersion.Excel2013;
//Do some manipulation
//Do some manipulation
//Workbook is saved in Excel2013 format
workbook.SaveAs("Sample.xlsx");
```

VB.NET

```
Dim excelEngine As New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
'Set application version
application.DefaultVersion = ExcelVersion.Excel2013
'Do some manipulation
'Do some manipulation
'Workbook is saved in Excel2013 format
workbook.SaveAs("Sample.xlsx")
```

If the application version is ignored, then the workbook version should be set properly during creation and save.

- To save a workbook in Excel2003 format, set the workbook version to Excel97to2003 and save the file with extension '.xls' i.e. binary file format.
- To save a workbook in Excel 2007 and above formats, set the workbook version to Excel2007 and above and save the file with extension '.xlsx' i.e. open XML file format.

These are represented in the below code snippet.

C#

```
workbook.Version = ExcelVersion.Excel97to2003;
workbook.SaveAs ("Sample.xls");
workbook.Version = ExcelVersion.Excel2013;
workbook.SaveAs ("Sample.xlsx");
```

VB.NET

```
workbook.Version = ExcelVersion.Excel97to2003
workbook.SaveAs ("Sample.xls")
workbook.Version = ExcelVersion.Excel2013
workbook.SaveAs ("Sample.xlsx")
```

What happens when an Excel file containing uninstalled fonts is converted to PDF/Image?

When the fonts used in particular Excel document are not installed in the machine, the desired characters will be missing in the PDF/Image conversion. However, XlsIO comes up with a font substitution method through **SubstituteFontEventHandler** event. This will enable user to specify alternate font name to render the characters in the specified alternate font. Otherwise, Microsoft Sans Serif is used as the default one.

Note: Due to this font substitution, there might be a slight difference with the rendered text in the generated PDF/Image files during Excel to PDF/Image conversion.

The following code snippet shows how to use font substitution in Excel to PDF conversion using XlsIO.

C#

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    IWorkbook workbook = application.Workbooks.Open("Sample.xlsx");
    //Initializes the SubstituteFont event to perform font substitution during Excel to PDF conversion
    application.SubstituteFont += new
        SubstituteFontEventHandler(SubstituteFont);
    ExcelToPdfConverter converter = new ExcelToPdfConverter(workbook);
    PdfDocument pdf = converter.Convert();
    Stream stream = File.Create("Output.pdf");
    pdf.Save(stream);
}

private static void SubstituteFont(object sender, SubstituteFontEventArgs
args)
{
    //Sets the alternate font when a specified font is not installed in the production environment
    if (args.OriginalFontName == "Wingdings Regular")
        args.AlternateFontName = "Bauhaus 93";
    else
        args.AlternateFontName = "Times New Roman";
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
```

```

Dim application As IApplication = excelEngine.Excel
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx")
'Initializes the SubstituteFont event to perform font substitution during
Excel to PDF conversion
AddHandler application.SubstituteFont, AddressOf SubstituteFont
Dim converter As ExcelToPdfConverter = New ExcelToPdfConverter(workbook)
Dim pdf As PdfDocument = converter.Convert()
Dim stream As Stream = File.Create("Output.pdf")
pdf.Save(stream)
End Using
Private Shared Sub SubstituteFont(ByVal sender As Object, ByVal args As
SubstituteFontEventArgs)
'Sets the alternate font when a specified font is not installed in the
production environment.
If args.OriginalFontName = "Wingdings Regular" Then
args.AlternateFontName = "Bauhaus 93"
Else
args.AlternateFontName = "Times New Roman"
End If
End Sub

```

How to avoid exception when adding worksheets with same name

Microsoft Excel throws exception when adding worksheet with existing worksheet name in a workbook and XlsIO does the same. But in some case, if you want to add worksheets with the same name using XlsIO then you can avoid the exception in XlsIO by setting IApplication IgnoreSheetNameException property as true.

The following code snippet shows how to add two worksheets with same name in a workbook.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Set IgnoreSheetNameException property as true
    application.IgnoreSheetNameException = true;
    //Create worksheets with same name
    IWorksheet sheet_1 = workbook.Worksheets.Create("Sheet");
    IWorksheet sheet_2 = workbook.Worksheets.Create("Sheet");
    string fileName = "Output.xlsx";
    workbook.SaveAs(fileName);
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2013
Dim workbook As IWorkbook = application.Workbooks.Create(1)
'Set IgnoreSheetNameException property as true
application.IgnoreSheetNameException = true
'Create worksheets with same name
Dim sheet_1 As IWorksheet = workbook.Worksheets.Create("Sheet")

```

```
Dim sheet_2 As IWorksheet = workbook.Worksheets.Create("Sheet")
Dim fileName As String = "Output.xlsx"
workbook.SaveAs(fileName)
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Set IgnoreSheetNameException property as true
    application.IgnoreSheetNameException = true;
    //Create worksheets with same name
    IWorksheet sheet_1 = workbook.Worksheets.Create("Sheet");
    IWorksheet sheet_2 = workbook.Worksheets.Create("Sheet");
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Output";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
});
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
    //Set IgnoreSheetNameException property as true
    application.IgnoreSheetNameException = true;
    //Create worksheets with same name
    IWorksheet sheet_1 = workbook.Worksheets.Create("Sheet");
    IWorksheet sheet_2 = workbook.Worksheets.Create("Sheet");
    //Saving the workbook as stream
    FileStream file = new FileStream("Output.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
    workbook.SaveAs(file);
    file.Dispose();
}
```

XAMARIN

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2013;
    IWorkbook workbook = application.Workbooks.Create(1);
```

```

//Set IgnoreSheetNameException property as true
application.IgnoreSheetNameException = true;
//Create worksheets with same name
IWorksheet sheet_1 = workbook.Worksheets.Create("Sheet");
IWorksheet sheet_2 = workbook.Worksheets.Create("Sheet");
//Saving the workbook as stream
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
stream.Position = 0;
//Save the document as file and view the saved document
//The operation in SaveAndView under Xamarin varies between Windows Phone,
//Android and iOS platforms. Please refer xlsio/xamarin section for respective
//code samples.
if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
TargetPlatform.Windows)
{
Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Output.xlsx", "application/msexcel", stream);
}
else
{
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Output.xlsx", "application/msexcel", stream);
}
}

```

How to change data point label color of a Waterfall chart

The following code snippet shows how to change data point label color of a Waterfall chart.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
application.DefaultVersion = ExcelVersion.Excel2016;
IWorkbook workbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic);
IWorksheet sheet = workbook.Worksheets[0];
//Accessing first chart in the sheet
IChartShape chart = sheet.Charts[0];
//Changing first data point label color
chart.Series[0].DataPoints[0].DataLabels.IsValue = true;
chart.Series[0].DataPoints[0].DataLabels.RGBColor = Color.Green;
workbook.SaveAs("Waterfall.xlsx");
}

```

VB.NET

```

Using excelEngine As ExcelEngine = New ExcelEngine()
Dim application As IApplication = excelEngine.Excel
application.DefaultVersion = ExcelVersion.Excel2016
Dim workbook As IWorkbook = application.Workbooks.Open("Sample.xlsx",
ExcelOpenType.Automatic)
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Create a chart
Dim chart As IChartShape = sheet.Charts(0)

```

```
'Changing first data point label color
chart.Series(0).DataPoints(0).DataLabels.IsValue = true
chart.Series(0).DataPoints(0).DataLabels.RGBColor = Color.Green
workbook.SaveAs("Waterfall.xlsx")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    //Instantiates the File Picker
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
    openPicker.FileTypeFilter.Add(".xlsx");
    openPicker.FileTypeFilter.Add(".xls");
    StorageFile file = await openPicker.PickSingleFileAsync();
    //Opens the workbook
    IWorkbook workbook = await application.Workbooks.OpenAsync(file,
        ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Accessing first chart in the sheet
    IChartShape chart = sheet.Charts[0];
    //Changing first data point label color
    chart.Series[0].DataPoints[0].DataLabels.IsValue = true;
    chart.Series[0].DataPoints[0].DataLabels.RGBColor = Color.Green;
    //Initializes FileSavePicker
    FileSavePicker savePicker = new FileSavePicker();
    savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
    savePicker.SuggestedFileName = "Waterfall";
    savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xlsx"
    });
    //Creates a storage file from FileSavePicker
    StorageFile storageFile = await savePicker.PickSaveFileAsync();
    //Saves changes to the specified storage file
    await workbook.SaveAsAsync(storageFile);
}
```

ASP.NET CORE

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    FileStream inputStream = new FileStream("Sample.xlsx", FileMode.Open,
        FileAccess.Read);
    IWorkbook workbook =
        application.Workbooks.Open(inputStream, ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    //Accessing first chart in the sheet
    IChartShape chart = sheet.Charts[0];
    //Changing first data point label color
    chart.Series[0].DataPoints[0].DataLabels.IsValue = true;
    chart.Series[0].DataPoints[0].DataLabels.RGBColor = Color.Green;
    //Saving the workbook as stream
```

```

FileStream stream = new FileStream("Waterfall.xlsx", FileMode.Create,
    FileAccess.ReadWrite);
workbook.SaveAs(stream);
stream.Dispose();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Excel2016;
    ///"App" is the class of Portable project
    Assembly assembly = typeof(App).GetTypeInfo().Assembly;
    Stream inputStream =
        assembly.GetManifestResourceStream("SampleBrowser.XlsIO.Samples.Template.Sample.xlsx");
    IWorkbook workbook =
        application.Workbooks.Open(inputStream, ExcelOpenType.Automatic);
    IWorksheet sheet = workbook.Worksheets[0];
    ///Accessing first chart in the sheet
    IChartShape chart = sheet.Charts[0];
    ///Changing first data point label color
    chart.Series[0].DataPoints[0].DataLabels.IsValue = true;
    chart.Series[0].DataPoints[0].DataLabels.RGBColor = Color.Green;
    ///Saving the workbook as stream
    MemoryStream stream = new MemoryStream();
    workbook.SaveAs(stream);
    stream.Position = 0;
    ///Save the document as file and view the saved document
    ///The operation in SaveAndView under Xamarin varies between Windows Phone,
    Android and iOS platforms. Please refer xlsio/xamarin section for respective
    code samples.
    if (Device.OS == TargetPlatform.WinPhone || Device.OS ==
        TargetPlatform.Windows)
    {
        Xamarin.Forms.DependencyService.Get<ISaveWindowsPhone>().SaveAndView("Waterfall.xlsx", "application/msexcel", stream);
    }
    else
    {
        Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("Waterfall.xlsx", "application/msexcel", stream);
    }
}

```

How to check whether an Excel document contains macro?

You can check whether the Excel document contains macro using **IWorkbook.HasMacro** property. The value true indicates that the Excel document has a Vba project.

The following code illustrates how to check whether an Excel document contains macro using XlsIO.

C#

```

using (ExcelEngine excelEngine = new ExcelEngine())
{

```



```
// Instantiate the excel application object.
IApplication application = excelEngine.Excel;
// Opening a workbook
IWorkbook workbook = application.Workbooks.Open("Test.xls");
IWorksheet sheet = workbook.Worksheets[0];
//Check macro exist
bool IsMacroEnabled = workbook.HasMacros;
//Save the workbook
workbook.SaveAs("Output.xls");
}
```

VB.NET

```
Using excelEngine As ExcelEngine = New ExcelEngine()
'Instantiate the excel application object.
Dim application As IApplication = excelEngine.Excel
'Opening a Workbook
Dim workbook As IWorkbook = application.Workbooks.Open("Test.xls")
Dim sheet As IWorksheet = workbook.Worksheets(0)
'Check macro exist
Dim IsMacroEnabled As Boolean = workbook.HasMacros
'Save the workbook
workbook.SaveAs("Output.xls")
End Using
```

UWP

```
using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//Instantiates the File Picker
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
openPicker.FileTypeFilter.Add(".xlsm");
openPicker.FileTypeFilter.Add(".xltn");
openPicker.FileTypeFilter.Add(".xls");
StorageFile file = await openPicker.PickSingleFileAsync();
//Opening a workbook with a worksheet
IWorkbook workbook = await application.Workbooks.OpenAsync(file,
ExcelOpenType.Automatic);
IWorksheet worksheet = workbook.Worksheets[0];
//Check macro exist
bool IsMacroEnabled = workbook.HasMacros;
// Save the Workbook
StorageFile storageFile;
if
(! (Windows.Foundation.Metadata.ApiInformation.IsTypePresent("Windows.Phone.UI.Input.HardwareButtons")))
{
FileSavePicker savePicker = new FileSavePicker();
savePicker.SuggestedStartLocation = PickerLocationId.Desktop;
savePicker.SuggestedFileName = "Output";
savePicker.FileTypeChoices.Add("Excel Files", new List<string>() { ".xls"
});
storageFile = await savePicker.PickSaveFileAsync();
}
}
```

```

else
{
StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;
storageFile = await local.CreateFileAsync("Output.xls",
CreationCollisionOption.ReplaceExisting);
}
//Saving the workbook
await workbook.SaveAsAsync(storageFile);
// Launch the saved file
await Windows.System.Launcher.LaunchFileAsync(storageFile);
}

```

ASP.NET CORE

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
//Instantiate the excel application object.
IApplication application = excelEngine.Excel;
//Opening form module existing workbook
FileStream input = new FileStream(DataPathBase + "Test.xls", FileMode.Open,
FileAccess.ReadWrite);
IWorkbook workbook = application.Workbooks.Open(input);
IWorksheet sheet = workbook.Worksheets[0];
//Check macro exist
bool IsMacroEnabled = workbook.HasMacros;
// Save the workbook
FileStream output = new FileStream("Output.xls", FileMode.Create,
FileAccess.ReadWrite);
workbook.SaveAs(output);
input.Close();
output.Close();
}

```

XAMARIN

```

using (ExcelEngine excelEngine = new ExcelEngine())
{
IApplication application = excelEngine.Excel;
//"App" is the class of Portable project
Assembly assembly = typeof(App).GetTypeInfo().Assembly;
Stream inputStream = assembly.GetManifestResourceStream("App.Test.xls");
//Opening the workbook
IWorkbook workbook = application.Workbooks.Open(inputStream);
IWorksheet worksheet = workbook.Worksheets[0];
//Check macro exist
bool IsMacroEnabled = workbook.HasMacros;
//Saving as Excel without macros
MemoryStream stream = new MemoryStream();
workbook.SaveAs(stream);
//Save the stream into XLSX file
Xamarin.Forms.DependencyService.Get<ISave>().SaveAndView("sample.xls",
"application/msexcel", stream);
}

```

Does XlsIO support password protected macro in the Excel documents?

Yes. XlsIO preserves the password protection for the macro in the Excel documents for resaving process. But, it does not have support for remove and modify the password.

Does XlsIO support Excel files with macros that are digitally signed?

Yes. XlsIO preserves the digital signature from the macro-enabled Excel document. But, it does not have support for remove and modify the signatures.