2.6.0 LiveCode for FM Release Notes

Release Notes

LiveCode for FM (2.6.0)

Tool

- The speed of processing solutions with a large number of tables has been improved.

Runtime

Functions

Average

- The `Average` function will no longer return '?' when used with more than one parameter where at least one field returns an error when fetched.

Count

- The `Count` function will no longer return '?' when used with more than one parameter where at least one field returns an error when fetched.

ExecuteSQL

- The `ExecuteSQL` function is now able to perform the following forms of SQL queries:
    - `SELECT <table>.<column> FROM <table> FETCH FIRST 1 ROWS ONLY`

JSONSetElement

- Using the `JSONSetElement` function to set the value of a index in an array which did not already exist will now work correctly, any indexes between the last used index and the new index will be created with a `null` value.
- Using the `JSONSetElement` function with an array index as a path when setting the value of an object will now produce an appropriate error.
- Using the `JSONSetElement` function with an object key as a path when setting the value of an array will now produice an appropriate error.
- Using the `JSONSetElement` function with an integer as a path on an empty array, empty object, or empty string will now always produce the correct result.

- The `JSONSetElement` function will no longer return objects with keys missing the initial quote character.

List

- The `List` function will no longer return '?' when used with more than one parameter where at least one field returns an error when fetched.

Max

- The `Max` function will no longer return '?' when used with more than one parameter where at least one field returns an error when fetched.

Min

- The `Min` function will no longer return '?' when used with more than one parameter where at least one field returns an error when fetched.

StDev

- The `StDev` function will no longer return '?' when used with more than one parameter where at least one field returns an error when fetched.

StDevP

- The `StDevP` function will no longer return '?' when used with more than one parameter where at least one field returns an error when fetched.

Sum

- The `Sum` function will no longer return '?' when used with more than one parameter where at least one field returns an error when fetched.

Variance

- The `Variance` function will no longer return '?' when used with more than one parameter where at least one field returns an error wher fetched.

VarianceP

- The `VarianceP` function will no longer return '?' when used with more than one parameter where at least one field returns an error when fetched.

Layout Objects

- Layout objects with invalid styles will no longer cause the app to exit.

Fields

- Added support for the `Autocomplete from Valuelist` option for drop-down fields.

Lines

- Line objects which are vertical and are both left and right anchored will no longer stretch when adjusting the window width.
- Line objects which are horizontal and are both top and bottom anchored will no longer stretch when adjusting the window height.

Popovers

- Buttons in portals in popovers will now work on initial click.

Portals

- Related fields in portals which are related to the portal's table will now not be focusable if there is no related record and the relationship does not allow creation.
- Related fields in portals which are related to the portal's table will now update the correct row when edited.
- Related fields in portals which are related to the portal's table will now ensure a record is present when edited, if the relationship allows creation.

Script Steps

Close Window

- Closing a window with an exit script trigger which uses the `Close Window` script step will no longer cause the app to exit.

Enter Preview Mode

- The `Scaling` print property will now be applied in Preview Mode.

Go to Related Record

- The `Go to Related Record` script step will now go to the correct record in the target layout when used from a portal row.
- The `Go to Related Record` script step will now select the correct set of records in the target layout when used from a portal row.
- Going to a related record in an external table will now go to the correct record.

Print

- On Android, margins will no longer be implicitly added to printed documents.

Print Setup

- The `Scaling` property is now implemented.

Save Records as PDF

- On Android, margins will no longer be implicitly added to documents.

Valuelists

- Valuelists created from a related field and using an external filter table will now be created correctly.

Feature Support

Layouts

Layouts are implemented.

The following layout triggers are not yet implemented:

- OnGestureTap
- OnExternalCommandReceived

Table view state is not supported.

The status bar, formatting bar and ruler are not yet supported.

Typing into the active field whilst it is not visible is not yet implemented.

On macOS, if the system preference 'always show scrollbars' option is enabled then the scrolling content area size is not yet correct.

Layout mode is not supported.

Files

Files and their basic commands and functions are implemented.

Each file in a compiled app corresponds to one solution file that was included in the LiveCode for FM project.

Files can reference other files, but the name used for the reference must always be the leaf (without extension) name of the solution file include in the app.

Each file can create multiple windows, and database access and script execution works across files.

User accounts are not currently implemented. The User Name is always 'System User' and the account name is always '[Full Access]'.

The OnFileAVPlayerChanged trigger is yet to be implemented.

Accounts, privilege sets and extended privileges are yet to be implemented.

Valuelists

Valuelists are almost completely implemented.

The order and uniqueness of values in non-custom valuelists is determined by sorting case insensitively using standard Unicode rules. Using the index language or the resort-language setting is yet to be implemented.

Menubar and Custom Menus

Not fully implemented.

The [Standard FileMaker Menuset] is partially implemented. Only the following menus and commmands are available:

- File:
  - Close
  - Print Setup
  - Print
  - Edit
  - Cut
  - Copy
  - Paste
  - Clear
  - Select All
  - Export Field Contents
- View

  - Browse Mode
  - Preview Mode
  - Find Mode
  - View as Form
  - View as List
  - Status Toolbar

- Insert

  - Picture
  - Audio/Video
  - PDF
  - File
  - Current Date
  - Current Time
  - Current Username

- Records

  - New Record
  - Duplicate Record
  - Delete Record
  - Delete All Records
  - Refresh Window
  - Show All Records
  - Show Omitted Only
  - Omit Record
  - Modify Last Find
  - Unsort
  - Next Record
  - Previous Record

- Requests

  - Add New Request
  - Duplicate Request
  - Delete Request
  - Nect Request
  - Previous Request
  - Revert Request

- Window

  - New Window
  - Minimize Window

The `View as Form` and `View as List` menu items are always enabled when shown in a menu.

The `Status Toolbar` menu item is enabled when toolbars are unlocked but will only affect the roolbar in Preview mode.

Some menu items and submenus which are not yet implemented are included in the standard menuset but are always disabled, i.e. `Go To Layout`, `Zoom In`, `Zoom Out`.

Menu items that are not supported are not included in menus, i.e. `File Options`, `Layout Mode`, `Manage Database`. Keyboard shortcu are implemented for all available menu items except Next Record/Request and Previous Record/Request as these conflict with system shortcuts on MacOS.

Custom menusets are partially implemented, with differences. The menu names "[File]", "[Edit]", "[View]", "[Insert]", "[Records]", and "[Requests]" cannot be used. In order to be displayed correctly in menu titles or menu item names the '/' character should be replaced with '//'. Script steps as menu items are fully implemented. Only the menu commands listed above are available.

Sync

Automatic and manual sync

Sync can be triggered automatically, or controlled manually, or both. If automatic sync is enabled, the application will block the UI and sync immediately on startup (before any scripts are run). This is a full sync, i.e. it includes checking for deletions.

Any local commit made while automatic sync is enabled causes a sync. This may or may not include checking for deletions, depending on a separate option. When automatic sync is enabled, remote server checks are scheduled to occur a number of seconds after the previous check according to the sync interval. The frequency of deletion checks included in the scheduled checks is configurable. When a server check is mad which does not include checking for deletions, this occurs in the background and no sync occurs (and hence no blocking of the UI occurs) if there is no new data to sync. When it is a full check, the UI is blocked immediately.

Manual sync is controlled using the extended script steps provided by the LiveCode for FM plugin. The `Sync` script step performs a sync immediately, blocking the UI. The `Request Sync` script step causes a check for changes at the next idle point.

If specific subsets of records are required by the app using the `Set Table Sync Constraint` script step, automatic sync must be turned on startup so that the constraints can be set before any sync occurs, since the automatic sync on startup would download all records from the server. Whether automatic sync is enabled or not can be controlled by the `Set Automatic Sync Enabled` extended script step.

Configuring Sync Solution

Configuring the sync solution settings can be done through two key script steps.

`Set Server Address` sets the FileMaker server address to the provided value, such as when needing to sync with interface files hosted on different servers.

`Set Sync Solution Name` sets the name of the interface file that your solution is synchronised with, and can be used in conjunction with `Set Server Address`, or independently.

Both clear all local data in syncing tables to ensure correct data is downloaded on the next sync from the new server / solution.

`Set Sync Solution Account` allows you to update the account details used when connecting to a sync solution, such as if the new sync solution set using `Set Sync Solution Name` has different account details.

Attempting to use either of these script steps when any records are locked will cause an error to be returned.

Deferred containers

When remote changes are detected that include container fields, the new container data is downloaded by default in the ensuing sync, causing the ui to block for however long this takes. This may cause unnecessary delays if the container data might not be needed immediately by the currently running app.

A container field can be marked as 'deferred', by adding the tag `@deferred` to the field's comment. Deferred containers are not downloaded during sync but instead downloaded when first used. If the container contents is required to display in a field object on a layout then this occurs asynchronously; if the container contents is required as part of a calculation, it will occur synchronously and the UI will be blocked.

If deferred container data is downloaded and stored which exceeds the current cache limit (initialized at 200Mb), then the next time a deferred container is resolved, data will be removed from storage to accommodate the new data. The amount of stored data after resolving a deferred container will therefore be either less than the cache limit, or the minimum required to display all visible containers on the most recent layout wi deferred containers.

Requirements

In order for a table to sync with LiveCode for FM's built in syncing capability, it must have the following three fields:

```
- a primary key field
- a modification timestamp field
- a creation timestamp field
```

When compiling a solution, LiveCode for FM will attempt to infer these fields from each basetable's definition. Specifically:

```
- The first field (in creation order) which has an autoenter calculation
  set to `Get(UUID)` is taken to be the primary key field.
- The first field (in creation order) which is set to be autoenter
  modification timestamp is taken to be the modification timestamp field.
- The first field (in creation order) which is set to be autoenter
  creation timestamp is taken to be the creation timestamp field.
```

> *Important:* All three fields must have either have a default index, or be set to auto-index. The primary key field must be modifiable. Without these settings sync will not function.

You can explicitly override the inferred fields by adding an appropriate tag to the relevant field's comment:

```
- @primary means LiveCode for FM will treat that field as the primary key
  field in preference to any other.
- @modification means LiveCode for FM will treat that field as the
  modification timestamp field in preference to any other.
- @creation means LiveCode for FM will treat that field as the creation
  timestamp field in preference to any other.
```

In each case, LiveCode for FM will take the first field (in creation order) it finds which has one of these tags. For the tag to be recognised it mus be surrounded by whitespace (or beginning / end of the comment). e.g.

`This field is the @primary key field`: correct `@primary`: correct `@primary key`: correct `Field @primary`: correct `This is the primary key field (@primary)`: not correct

> *Note:* If you tag fields in this way, then the only restriction LiveCode for FM puts on their settings is that they must be a stored normal field (global, calculated, and summary fields are ignored).

Layout Objects

The majority of layout objects are implemented.

Layout object tooltips are not yet implemented.

The following layout object triggers are not yet implemented:

- OnObjectAVPlayerChange

Button

Button layout objects are implemented.

Button Bar

Button bar layout objects are implemented.

Button Bar Button

Button bar button layout objects are implemented.

Button Bar Popover Button

Button bar popover button layout objects are implemented.

Calendar Field

Calendar field layout objects are implemented.

Autocomplete is not currently supported.

Check Box Field

Check Box field layout objects are implemented.

Container Field

Container field layout objects are implemented.

The container field layout object supports GIF, JPEG, PNG and BMP format images.

The container field layout object supports MP3, MP4 and WAV audio and video formats.

The container field layout object supports PDF files, although currently only the first page is shown and they are not pannable.

The container field layout object supports general file content.

Edit Box Field

Edit field layout objects are implemented.

Autocomplete is not currently supported.

Group Button

Group button layout objects are implemented.

Currently children of a group button object will not highlight whilst the mouse is pressed on the group button.

Image

Image layout objects are implemented.

Line

Line layout objects are implemented.

List Field

List field layout objects are implemented.

The Autocomplete option is not yet implemented.

Note: List field objects which were created in historic versions of FileMaker and not changed in more recent versions can loose their 'Show Icor property. To rectify this, toggle the property value in the FileMaker UI and re-run the import phase.

Menu Field

Menu field layout objects are implemented.

Note: Menu field objects which were created in historic versions of FileMaker and not changed in more recent versions can loose their 'Show Icon' property. To rectify this, toggle the property value in the FileMaker UI and re-run the import phase.

Oval

Oval layout objects are implemented.

Popover

Popover layout objects are implemented.

Popover Button

Popover button layout objects are implemented.

Portal

Portals are implemented.

The 'retain scrollbar position' option is not yet implemented.

The 'allow row deletion' option is not yet implemented.

Radio Field

Radio field layout objects are implemented.

Rectangle

Rectangle layout objects are implemented.

Rounded Rectangle

Rounded rectangle layout objects are implemented.

Slide Control

Slide control layout objects are implemented.

Tab Control

Tab control layout objects are implemented.

Text

Text layout objects are implemented.

Webviewer

Webviewer layout objects are implemented.

Custom url schemes are supported, and you can specify the schemes which should be handled at runtime when compiling a project. Only solution files compiled into the app can be used. Use either the `$` or `~` form of custom urls, and use the file name of the solution as provided t the compiler.

Not Yet Implemented

- Chart

Layout Parts

Layout parts are implemented.

Printing Implementation Details

Printing of simple print-oriented layouts is implemented.

The `Enter Preview Mode` script step works on all platforms, and has an initial prototype user interface allowing exit, page navigation, savin as pdf, page setup and print.

The `Print` script step works on all platforms and allows printing a layout through the system printing user interface.

The print-related part options are implemented.

The `Do not print`, `Sliding left` and `Sliding up` print-related layout object options are implemented.

Printing of the following layout objects is implemented:

- Field objects
- Button objects
- Button bar objects
- Rectangle objects
- Rounded Rectangle objects
- Grouped objects
- Oval objects
- Line objects
- Image objects
- Portal objects

Printing a blank record as formatted, with blank boxes, with placeholders and underlined is implemented.

The `Delineate fields on current record only` layout option is implemented.

The evaluation of hide conditions while printing is implemented.

The evaluation of conditional formatting while printing is implemented.

The majority of style related options is implemented. Notable exceptions are:

- full justification of text
- text tab stops
- image fills

Tables, Fields and Relationships

Tables and fields are almost completely implemented.

Field validation is yet to be implemented.

Table, record and field level access are yet to be implemented.

Find Request Query Operators

Find request query operators are partially implemented.

The following operators are available:

- is empty: `=`
- is not empty: `*`
- is less than or equal: `<=<value>` or `≤<value>`
- is greater than or equal: `>=<value>` or `≥<value>`
- is less than: `<<value>`
- is greater than: `><value>`
- ranges: `<value>…<value>`, `<value>..<value>` or `<value>...<value>`
- is equal to: `<value>` or `=<value>` or `==<value>` or `"<value>"` (when the datatype is not text)
- is exactly: `==<value>` (when the datatype is text)
- matches patterns using `*` as wildcards
- contains word: `=<value>` (when the datatype is text)
- contains word beginning with: `<value>` (when the datatype is text)
- contains phrase beginning with: `"<value>"` (when the datatype is text)

Find request criteria using `>`, `>=`, `≥`, `<`, `<=` and `≤` may have spaces between the operator and value.

Functions

Not Currently Supported

Context

GetAllowFormattingBarState

Not currently supported. Returns 0.

GetStatusAreaState

Not currently supported. Returns 0.

GetTextRulerVisible

Not currently supported. Returns 0.

Implemented With Differences

Container

CryptAuthCode

Implemented with differences. The SHA and MDC2 digest algorithms are not supported.

CryptDigest

Implemented with differences. The SHA and MDC2 digest algorithms are not supported.

Context

GetAccountExtendedPrivileges

Implemented as override. Currently the value returned by Get(AccountExtendedPrivileges) will be that specified by the corresponding override at compile-time, or empty if one isn't set.

GetAccountGroupName

Implemented as override. Currently the value returned by Get(AccountGroupName) will be that specified by the corresponding override at compile-time, or empty if one isn't set.

GetAccountName

Implemented as override. Currently the value returned by Get(AccountName) will be that specified by the corresponding override at compile-time, or

GetAccountPrivilegeSetName

Implemented as override. Currently the value returned by Get(AccountPrivilegeSetName) will be that specified by the corresponding override a compile-time, or "[Full Access]" if one isn't set.

GetAccountType

Implemented as override. Currently the value returned by Get(AccountType) will be that specified by the corresponding override at compile-time, or "FileMaker File" if one isn't set.

GetApplicationLanguage

Implemented with differences. Currently the value returned by Get(ApplicationLanguage) is always "English".

GetApplicationVersion

Implemented with override. The value returned by Get(ApplicationVersion) will be that specified as the corresponding override at compile-time. If no override is set then "Go 18.0.0" is returned.

GetConnectionAttributes

Implemented with differences. Solutions run locally and are not hosted so the return value is always empty.

GetConnectionState

Implemented with differences. Solutions run locally and are not hosted so the return value is always 0.

GetCurrentPrivilegeSetName

Implemented with differences. The value returned by Get(CurrentPrivilegeSetName) is the same as Get(AccountPrivilegeSetName).

GetDevice

Implemented with override. The value returned by Get(Device) will be that specified as the corresponding override at compile-time. If no override is set then it returns the same value as FileMaker for the current platform.

GetHostApplicationVersion

Implemented with differences. Solutions run locally and are not hosted so the return value is always empty.

GetHostIPAddress

Implemented with differences. Solutions run locally and are not hosted so the return value is always empty.

GetMultiUserState

Implemented with differences. Solutions run locally and are not hosted so the return value is always 0.

GetNetworkType

Implemented with differences. Solutions run locally and are not hosted so the return value is always 0.

GetPersistentID

Implemented with differences. The id returned is generated on first-run of the app on a given device. The id will survive backups but not necessarily deletion and reinstallation of the app.

GetSystemPlatform

Implemented with override. The value returned by Get(SystemPlatform) will be that specified as the corresponding override at compile-time. If no override is set then it returns the same value as FileMaker for the current platform.

GetUserCount

Implemented with differences. Solutions run locally and are not hosted so the return value is always 1.

GetUserName

Implemented with override. The value returned by Get(UserName) will be that specified as the corresponding override at compile-time. If no override is set then it returns the full name of the current user on macOS, `iPad` or `iPhone` on iOS, and `Android` on android.

GetWindowOrientation

Implemented with override. The value returned by Get(WindowOrientation) will be that specified as the corresponding override at compile-time. If no override is set then it returns the same value as FileMaker would.

Design

WindowNames

Implemented with differences. The list of window names for a given file are in order of most recently opened or brought to the front. If no file name is specified then the windows are grouped by file in the list.

Logical

Evaluate

Implemented with differences. If the expression string fails to parse then the evaluation error will be set to 1200 (generic calculation error), rath
than a more specific code.

Mobile

Location

Implemented with differences. Implemented on mobile platforms only. Allowing users to cancel the location fetching process when Get(AllowAbortState) is true is yet to be implemented.

LocationValues

Implemented with differences. Implemented on mobile platforms only. Allowing users to cancel the location fetching process when Get(AllowAbortState) is true is yet to be implemented.

Not Fully Implemented

Container

GetContainerAttribute

Not fully implemented. The following groups are mostly implemented:

- all
- general
- image
- photo
- barcode

The following attributes are implemented:

- filename
- fileSize
- width
- height
- dpiWidth
- dpiHeight
- orientation
- make
- model

- created
- modified
- latitude
- longitude
- barcodeText
- barcodeType

Logical

GetAVPlayerAttribute

Not fully implemented. The following options are not yet implemented.

- `triggerEvent`
- `triggerEventDetail`
- `sequence`
- `disableExternalControls`
- `pauseInBackground`
- `pictureInPicture`
- `externalPlayback`
- `imageSourceType`
- `imageSource`
- `imageDuration`

GetLayoutObjectAttribute

Not fully implemented. Only the following attributes can be used:

- objectType
- hasFocus
- isFrontPanel
- isFrontTabPanel
- isObjectHidden
- containedObjects
- rotation

IsValid

Not fully implemented. The ability to check whether a field's content matches its data-type is not yet implemented.

Mobile

GetSensor

Not fully implemented. Only `location`, `locationValues` and `available` are implemented.

Not Yet Implemented

Aggregate

- Last

Container

- VerifyContainer

Context

- GetCurrentExtendedPrivileges
- GetEncryptionState
- GetHighConstrastState
- GetHighContrastColor
- GetLayoutAccess
- GetMenubarState
- GetPreferencesPath
- GetPrinterName
- GetQuickFindText
- GetRecordAccess
- GetRegionMonitorEvents
- GetScreenScaleFactor
- GetScriptAnimationState
- GetTriggerExternalEvent
- GetTriggerGestureInfo
- GetWindowZoomLevel

Logical

- LookupNext

Mobile

- RangeBeacons

Number

- Combination
- Factorial

Text

- Furigana
- Hiragana
- KanaHankaku
- KanaZenkaku
- KanjiNumeral
- Katakana
- NumToJText
- RomanHankaku
- RomanZenkaku

Not Supported

Container

- CryptDecrypt
- CryptDecryptBase64
- CryptEncrypt
- CryptEncryptBase64

Context

- GetInstalledFMPlugins

Script Steps

Not Currently Supported

Miscellaneous

Allow Formatting Bar

Not currently supported. Has no effect.

Windows

Show Hide Text Ruler

Not currently supported. Has no effect.

Remote operations

A number of script steps can only be run on the server. Currently this collection consists of the account-related script steps and Perform Script On Server. The steps are executed by running particular scripts on the server after logging in via the data API. As such, in order for them to wo
there are some application prerequisites:

1. The solution must be hosted on FileMaker Server
2. Any account under which the script steps are intended to be run must have the fmrest privilege
3. The target script must exist in the hosted solution.

   Important: As they are executed via the Data API, the bespoke scripts must be uniquely named, otherwise the correct script may not be called, even if the scripts are in different folders.

In the case of the account-related script steps, with the exception of Re-Login, the scripts in question should be of a specific form:

```
# LiveCode for FM <Operation Name> Script

Set Error Capture [ On ]
<Run account script step>
Exit Script [ Text Result: Get(LastError) ]
```

Examples for each step can be seen below.

Add Account

The Add Account remote script must contain the comment

```
# LiveCode for FM Add Account Script
```

somewhere within the script. The parameters passed when executing this script are:

1. Expire password
2. Account name
3. Password
4. Privilege set name

The implementation itself should not use the 'User must change password on next sign-in' (Expire password) option, as it is not supported. Since the privilege set cannot be a calculation, each case must be hard-coded in an If ... Else If ... block.

Example:

```
# LiveCode for FM Add Account Script

Set Error Capture [ On ]

# The expire password variable is not used. If it is true, return
# "invalid-command".
Set Variable [ $expire_password ; Value: GetValue(Get(ScriptParameter);1) ]
If [ $expire_password ]
    Exit Script [ Text Result: 3 ]
End If

Set Variable [ $username ; Value: GetValue(Get(ScriptParameter);2) ]
Set Variable [ $password ; Value: GetValue(Get(ScriptParameter);3) ]
Set Variable [ $privilege_set ; Value: GetValue(Get(ScriptParameter);4) ]

If [ $privilege_set = "[Data Entry Only]" ]
    Add Account [ Account Name: $username ; Password : $password ;
    Privilege Set: [Data Entry Only] ]
Else If [ $privilege_set = "[Read-Only Access]" ]
    Add Account [ Account Name: $username ; Password : $password ;
    Privilege Set: [Read-Only Access] ]
Else
    # Handle custom privilege sets
    If [ $privilege_set = "My Custom Privilege Set" ]
        Add Account [ Account Name: $username ; Password : $password ;
        Privilege Set: My Custom Privilege Set ]
    Else
        # Privilege set does not exist
        Exit Script [ Text Result: 3 ]
    End if
End If
Exit Script [ Text Result: Get(LastError) ]
```

Change Password

The Change Password remote script must contain the comment

```
# LiveCode for FM Change Password Script
```

somewhere within the script. The parameters passed when executing this script are:

1. Current password
2. New password

Example:

```
# LiveCode for FM Change Password Script

Set Error Capture [ On ]

Set Variable [ $old_password ; Value: GetValue(Get(ScriptParameter);1) ]
Set Variable [ $new_password ; Value: GetValue(Get(ScriptParameter);2) ]

Change Password [ Old Password: $old_password ; Password : $new_password
With dialog: Off ]
Exit Script [ Text Result: Get(LastError) ]
```

Configuring Server Address

This changes the server on which the remote operations are performed.

If sync is enabled, then all local data in syncing tables will be cleared to ensure the correct data is downloaded on next sync.

Attempting to use this script step when any records are locked will cause an error to be returned.

Example:

```
# Validate user in a central server solution, fetch server address,
# change the server address to the user's server and sync data.

# Log into a user account in a solution hosted on original server
Re-Login [ Account Name: gUserName ; Password: ******** ; With dialog: Off ]
If [ Get(LastError) ≠ 0 ]
    Exit Script [ Text Result: "failed to login" ]
End If
# Run script which returns appropriate server address based on the value of
# Get(AccountName)
Perform Script on Server [ Specified: From list ; "Fetch User Server Address" ; Parameter:   ; Wait for
completion: On ]
Set Variable [ $server_address ; Value: Get(ScriptResult) ]

# Set the new server address
Set Server Address [ Address: $server_address ]

# Run script on new server which returns user id
Perform Script on Server [ Specified: From list ; "Fetch User Id" ; Parameter:   ; Wait for completion:
On ]
Set Variable [ $user_id ; Value: Get(ScriptResult) ]

# Constrain data table(s) and sync
Set Table Sync Constraint [ Table: "UserData" ; Constraint: "\"fkUserId\"=' & $user_id & '" ]
Sync [ Check For Deletions: 1; Timeout:   ]
```

Delete Account

The Delete Account remote script must contain the comment

```
# LiveCode for FM Delete Account Script
```

somewhere within the script. The parameters passed when executing this script are:

1. Account name

Example:

```
# LiveCode for FM Delete Account Script

Set Error Capture [ On ]

Set Variable [ $account_name ; Value: GetValue(Get(ScriptParameter);1) ]

Delete Account [ Account Name: $account_name ]
Exit Script [ Text Result: Get(LastError) ]
```

Enable Account

The Enable Account remote script must contain the comment

```
# LiveCode for FM Enable Account Script
```

somewhere within the script. The parameters passed when executing this script are:

1. Activate (true or false)
2. Account name

Since whether to activate or deactivate cannot be a calculation, each case must be hard-coded in an If ... Else ... block.

Example:

```
# LiveCode for FM Enable Account Script

Set Error Capture [ On ]

Set Variable [ $is_activate ; Value: GetValue(Get(ScriptParameter);1) ]
Set Variable [ $account_name ; Value: GetValue(Get(ScriptParameter);2) ]

If [ $is_activate ]
    Enable Account [ Account Name: $account_name ; Activate ]
Else
    Enable Account [ Account Name: $account_name ; Deactivate ]
End If
Exit Script [ Text Result: Get(LastError) ]
```

Re-Login

The Re-Login script step can be run without a remote script implementing it. In this case, the credentials are simply used to authorize Data API login. The script step can thus be used to verify that the account exists and the password is correct. In this case no local state is changed.

A remote script for Re-Login is needed in order to have the account related Get functions behave correctly after relogin.

The Re-Login remote script must contain the comment

```
# LiveCode for FM Re-Login Script
```

somewhere within the script. The parameters passed when executing this script are:

1. Account name
2. Password

The Re-Login remote script *must* return the account data as a carriage-return-delimited list in the correct order:

1. Last error code
2. Account type
3. Account name
4. Account group name
5. Account privilege set name

6. Account extended privileges

   Important: If the text result of this script differs from the above-specified format, the application may subsequently exhibit undesired behavior.

   Note: Since the provided credentials are used to log in to the Data API, it is not necessary for the Re-Login script to actually run the Re-Login script step. The key function of the script is to return the relevant account data.

Example:

```
# LiveCode for FM Re-Login Script

Exit Script [ Text Result: 0 & ¶ & Get(AccountType)
     & ¶ & Get(AccountName) & ¶ & Get(AccountGroupName)
     & ¶ & Get(AccountPrivilegeSetName) & ¶ & Get(AccountExtendedPrivileges) ]
```

Reset Account Password

The Reset Account Password remote script must contain the comment

```
# LiveCode for FM Reset Account Password Script
```

somewhere within the script. The parameters passed when executing this script are:

1. Expire password
2. Account name
3. Password

The implementation itself should not use the 'User must change password on next sign-in' (Expire password) option, as it is not supported.

Example:

```
# LiveCode for FM Reset Account Password Script

Set Error Capture [ On ]

# The expire password variable is not used. If it is true, return
# "invalid-command".
Set Variable [ $expire_password ; Value: GetValue(Get(ScriptParameter);1) ]
If [ $expire_password ]
    Exit Script [ Text Result: 3 ]
End If

Set Variable [ $account_name ; Value: GetValue(Get(ScriptParameter);2) ]
Set Variable [ $password ; Value: GetValue(Get(ScriptParameter);3) ]

Reset Account Password [ Account Name: $account_name ; Password : $password ]
Exit Script [ Text Result: Get(LastError) ]
```

Implemented

Fields

Insert From Device Bar Code From Camera

Implemented on mobile platforms only.

Insert From Device Music Library

Implemented on mobile platforms only.

Insert From Device Photo Library

Implemented on mobile platforms only.

Implemented With Differences

Account

Add Account

Implemented with differences. This script step can only be run on the server. See the 'Remote Operations' section for details. The 'User must change password on next sign-in' option is not supported.

Change Password

Implemented with differences. The 'with dialog' form is not yet implemented. This script step can only be run on the server. See the 'Remote Operations' section for details.

Delete Account

Implemented with differences. This script step can only be run on the server. See the 'Remote Operations' section for details.

Enable Account

Implemented with differences. This script step can only be run on the server. See the 'Remote Operations' section for details.

Re Login

Implemented with differences. The 'with dialog' form is not yet implemented. This script step can only be run on the server. See the 'Remote Operations' section for details.

Reset Account Password

Implemented with differences. This script step can only be run on the server. See the 'Remote Operations' section for details. The 'User must change password on next sign-in' option is not supported.

Control

Allow User Abort

Implemented with differences. The AllowUserAbort state can be set and retrieved with Get(AllowAbortState), but all script steps run as if it is se to False.

Set Error Capture

Implemented with differences. The ErrorCapture state can be set and retrieved with Get(ErrorCaptureState), but all script steps run as if it is se to True.

Fields

Export Field Contents

Implemented with differences. On desktop platforms, the 'Auto Open' and 'Create Email' options are not available.

Export Field Contents To File

Implemented with differences. On desktop platforms, the 'Create Email' option has no effect.

Insert From Device Bar Code From Field

Implemented with differences. Implemented on android platform only.

Files

Close File

Implemented with differences. Closing a file in an app closes all its windows and reverts it to being in the background.

Open File

Implemented with differences. All files compiled into an app are opened in the background on startup, calling Open File causes the file to be brought to the foreground, running appropriate triggers.

Note: You can only open files which were included in the app at compile-time.

Set Multi User

Implemented with differences. Solutions run locally and cannot be shared so this has no effect.

Menus

Install Menu Set

Implemented with differences. The menu names [File], [Edit], [View], [Insert], [Records], and [Requests] cannot be used. In order to be displayed correctly in menu titles or menu item names the '/' character should be replaced with '//'.

Miscellaneous

Configure Local Notification

Implemented with differences. Buttons on the notifications are not yet supported and in the script parameter line two will always be `NotificationSelected` and line 3 will always be 0 as these values are not yet being determined.

Navigation

Go To Layout

Implemented with differences. The `Flip from Left` and `Flip from Right` animations are only supported on iOS. On other platforms, they behave the same as `Slide to Right` and `Slide to Left` respectively.

Go To Related Record

Implemented with differences. The `Flip from Left` and `Flip from Right` animations are only supported on iOS. On other platforms, they behave the same as `Slide to Right` and `Slide to Left` respectively.

Records

Commit Records Requests

Implemented with differences. The skip-validation option currently has no effect as field validation is not yet implemented. The force-commit option is not supported.

Windows

Refresh Window

Implemented with differences. The flush-sql option is not supported.

Show Hide Toolbars

Implemented with differences. Only Preview Mode is supported. Include Edit Record Toolbar is not supported.

View As

Implemented with differences. The table view mode is not supported.

Not Fully Implemented

ExportRecords

Export Records

Not fully implemented. Only exporting without dialog to tab or comma-separated values is implemented. Exporting to FileMaker files is not supported.

Fields

Replace Field Contents With Calculation

Not fully implemented. The 'with dialog' form is not yet implemented.

Replace Field Contents With Current Contents

Not fully implemented. The 'with dialog' form is not yet implemented.

Replace Field Contents With Serial Numbers

Not fully implemented. The 'with dialog' form is not yet implemented.

Foundsets

Omit Multiple Records

Not fully implemented. The 'with dialog' form is not yet implemented.

Sort Records

Not fully implemented. The 'with dialog' form is not yet implemented.

ImportRecords

Import Records

Not fully implemented. Only importing into existing tables without dialog from tab or comma separated values is implemented. Importing from FileMaker files is not supported.

Miscellaneous

AVPlayer Play Field

Not fully implemented. `Full Screen Only` and `Embedded Only` presentation modes are not yet supported. Presentation will revert to `Start Full Screen` and `Start Embedded` respectively when these modes are requested. When playing a video in full screen with `Hide Controls` set to false, the user will not be able to exit from full screen mode during video playback.

AVPlayer Play Object

Not fully implemented. `Full Screen Only` and `Embedded Only` presentation modes are not yet supported. Presentation will revert to `Start Full Screen` and `Start Embedded` respectively when these modes are requested. When playing a video in full screen with `Hide Controls` set to false, the user will not be able to exit from full screen mode during video playback.

AVPlayer Play URL

Not fully implemented. `Full Screen Only` and `Embedded Only` presentation modes are not yet supported. Presentation will revert to `Start Full Screen` and `Start Embedded` respectively when these modes are requested. When playing a video in full screen with `Hide Controls` set to false, the user will not be able to exit from full screen mode during video playback.

AVPlayer Set Options

Not fully implemented. The following options are not yet implemented:

- Disable External Controls
- Pause In Background
- Media sequence
- Zoom modes `FitOnly`, `FillOnly` and `StretchOnly`.
- All zoom modes on Android When playing a video in full screen with `Hide Controls` set to false, the user will not be able to exit from full screen mode during video playback.

Navigation

Enter Preview Mode

Not fully implemented. See `Printing Implementation Details` section for more details.

Go To Record Request Page Number

Not fully implemented. The 'with dialog' form is not yet implemented.

Go To Related Record In New Window

Not fully implemented. Only 'document' and 'card' styles are supported. Only the 'name', 'x', 'y', 'width' and 'height' new window options are honoured.

Print

Print

Not fully implemented. The 'with dialog' form is not yet implemented. See the `Printing Implementation Details` section for further details.

Records

Delete All Records

Not fully implemented. The 'with dialog' form is not yet implemented.

Duplicate Record Request

Not fully implemented. Duplicating a record in the context of a non master-detail portal is yet to be implemented.

Windows

Adjust Window

Not fully implemented. The Restore option is currently unimplemented.

New Window

Not fully implemented. Only 'document' and 'card' styles are supported. Only the 'name', 'x', 'y', 'width', 'height', 'dim parent window', and 'toolbars' new window options are honoured.

Not Yet Implemented

Control

- Set Error Logging

Editing

- Perform Find Replace

- Undo Redo

Fields

- Insert File Into Variable
- Insert From Index
- Insert From Last Visited
- Relookup Field Contents

Files

- Convert File
- Open File With Dialog

Miscellaneous

- Clear Region Monitor Script
- Configure Region Monitor Script Geolocation
- Configure Region Monitor Script Ibeacon
- Get Folder Path
- Perform Apple Script
- Send DDEExecute File
- Send DDEExecute Topic
- Send Event With File
- Send Event With Message
- Set Layout Object Animation

Records

- Copy All Records Requests
- Copy Record Request
- Save Records As Excel
- Save Records As Snapshot Link

Spelling

- Check Current Field Selection
- Check Found Set
- Check Record
- Check Selection
- Correct Word
- Edit User Dictionary
- Select Dictionaries
- Spelling Options

Windows

- Arrange All Windows
- Set Zoom Level
- Show Hide Menubar

Not Supported

Fields

- Insert Object
- Update Link

Files

- Close ODBCData Source
- New File
- Open ODBCData Source
- Recover File
- Save ACopy As
- Save ACopy As To File
- Save ACopy As XML

Menus

- Open Edit Saved Finds
- Open Favorites
- Open File Options
- Open Find Replace
- Open Help
- Open Hosts
- Open Manage Containers
- Open Manage Data Sources
- Open Manage Database
- Open Manage Layouts
- Open Manage Themes
- Open Manage Value Lists
- Open Preferences
- Open Script Workspace
- Open Sharing
- Open Upload To Host

Miscellaneous

- Execute SQL
- Install Plug In File From Field

Windows

- Set Allowed Orientations

Extended Script Steps

Deferred container configuration

The LiveCode for FM plugin provides the ability to control the application's deferred container cache within FileMaker solution scripts. The current cache size can be obtained using the `GetDeferredContainerCacheSize` function, and the cache limit can be fetched and set usin the `GetDeferredContainerCacheLimit` function and the `Set Deferred Container Cache Limit` script step respectively.

Functions

GetAutomaticFullSyncFrequency

Purpose

Get the frequency of automatic full syncs

Format

```
GetAutomaticFullSyncFrequency
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Returns the number of normal sync intervals before a full sync. If the app was compiled without sync support, then `0` is returned.

Related topics

Set Automatic Sync Interval script step

GetAutomaticFullSyncOnLocalCommit

Purpose

Determine whether automatic sync is full or not after local commit

Format

```
GetAutomaticFullSyncOnLocalCommit
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Returns the value of the 'full sync on local commit' option, which determines whether deletion checks are performed when a sync is triggered b a local commit. If the app was compiled without sync support, then `False` is returned.

Related topics

Set Automatic Full Sync On Local Commit script step

GetAutomaticSyncEnabled

Purpose

Determine whether automatic sync is enabled or not.

Format

```
GetAutomaticSyncEnabled
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Determines whether the automatic syncing mechanism is enabled or not. If the app was compiled without sync support, then `False` is returned.

Example 1

Ensure sync happens whether automatic or not

```
If [ not GetAutomaticSyncEnabled() ]
        Request Sync
End If
```

Related topics

Set Automatic Sync Enabled script step

GetAutomaticSyncInterval

Purpose

Get the interval between automatic syncs

Format

```
GetAutomaticSyncInterval
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Returns the interval between checks for updates on the server, when there have been no local changes. The interval is in seconds. An interval 0 means sync will only happen when a local commit needs to be pushed. If the app was compiled without sync support, then `0` is returned.

Related topics

Set Automatic Sync Interval script step

GetDeferredContainerCacheLimit

Purpose

Return the deferred container cache limit

Format

```
GetDeferredContainerCacheLimit
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 2.0

Description

When deferred containers are resolved, they are cached. Use this function to return the current cache limit, in bytes.

Related topics

Set Deferred Container Cache Limit script step GetDeferredContainerCacheSize function

GetDeferredContainerCacheSize

Purpose

Return the deferred container cache size

Format

```
GetDeferredContainerCacheSize
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 2.0

Description

When deferred containers are resolved, they are cached. Use this function to return the current cache size, in bytes.

Related topics

Set Deferred Container Cache Limit script step GetDeferredContainerCacheLimit function

GetLastSyncTimeUTCMilliseconds

Purpose

Get the time of the last sync

Format

```
GetLastSyncTimeUTCMilliseconds
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Returns the time at which the last sync occurred in milliseconds since 1/1/0001 - i.e. it is comparable with `Get(CurrentTimeUTCMilliseconds).` If the app was compiled without sync support, then `0` is returned.

Related topics

Set Automatic Full Sync On Local Commit script step

GetSyncCompletionTimeout

Purpose

Return the current sync completion timeout

Format

```
GetSyncCompletionTimeout
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 2.0

Description

The amount of time to try and complete sync before aborting.

Related topics

Set Sync Completion Timeout script step

GetSyncSolutionAccountName

Purpose

Returns the FileMaker Sync solution account username

Format

```
GetSyncSolutionAccountName
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns the FileMaker Sync solution account username.

Related topics

Set Sync Solution Account script step

GetSyncSolutionName

Purpose

Returns the FileMaker Sync solution name

Format

```
GetSyncSolutionName
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes

Where the script step runs	Supported
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns the FileMaker Sync solution name.

GetTableSyncConstraint

Purpose

Get the sync constraint applied to a table

Format

```
GetTableSyncConstraint
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Returns the sync constraint that is applied to a table, previously set using the `Set Table Sync Constraint` script step. If the app was compiled without sync support, then empty is returned. type: calc label: Table datatype: Text showinline: true description: | The name of the tabl

Related topics

Set Table Sync Constraint script step

GetTableSyncMode

Purpose

Get the sync mode of a table

Format

```
GetTableSyncMode
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Script Steps

Cancel Sync

Purpose

Cancel a pending sync request.

Format

```
Cancel Sync
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Cancels any pending sync request that might be present. When automatic syncing is enabled, `Cancel Sync` cancels the next sync whether be scheduled according to the automatic sync interval or after a local commit. If automatic sync is disabled, this cancels a pending sync requested using the `Request Sync` script step. If the app was compiled without sync support, then `Get(LastError)` returns 65536 (Sync not supported).

Example 1

Prevent a local commit from being synced automatically

```
New Record/Request
Commit Records/Requests [ With dialog: Off ]
Cancel Sync
```

Example 2

Cancel an explicitly requested sync

```
Request Sync
Cancel Sync
```

Related topics

Request Sync script step

Delete All Local Records

Purpose

Delete all local records

Format

```
Delete All Local Records
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Planned but not yet implemented.

Delete Local Record

Purpose

Delete a local record

Format

```
Delete Local Record
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Planned but not yet implemented.

Disable Automatic Sync

Purpose

Disable automatic sync.

Format

```
Disable Automatic Sync
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Disables the automatic syncing mechanism. When automatic sync is disabled, no automatic syncing occurs at all - even if local changes are made. Sync is only triggered by the `Request Sync` and `Sync` script steps. If the app was compiled without sync support, then `Get(LastError)` returns 65536 (Sync not supported).

Example 1

Prevent a local commit from being synced automatically

```
Disable Automatic Sync
New Record/Request
Commit Records/Requests [ With dialog: Off ]
```

Related topics

Enable Automatic Sync script step GetAutomaticSyncIsEnabled function

Enable Automatic Sync

Purpose

Enable automatic sync.

Format

```
Enable Automatic Sync
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Enables the automatic syncing mechanism. When automatic sync is enabled, sync is triggered by local commits and also periodically accordin to the 'automatic sync interval' property. If the app was compiled without sync support, then `Get(LastError)` returns 65536 (Sync not supported).

Example 1

Sync a local commit as soon as possible

```
Enable Automatic Sync
New Record/Request
Commit Records/Requests [ With dialog: Off ]
```

Related topics

Disable Automatic Sync script step GetAutomaticSyncIsEnabled function

Omit All Journaled Records

Purpose

Omit all journaled records

Format

```
Omit All Journaled Records
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Planned but not yet implemented.

Request Sync

Purpose

Request a sync.

Format

```
Request Sync
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Requests a sync which will be performed as soon as possible. Requested syncs can be cancelled using the `Cancel Sync` script step. If the app was compiled without sync support, then `Get(LastError)` returns 65536 (Sync not supported).

Example 1

Trigger a sync with a particular local commit, but not all.

```
Disable Automatic Sync
New Record/Request
Commit Records/Requests [ With dialog: Off ]
Request Sync
```

Related topics

Cancel Sync script step

Resync All Tables

Purpose

Resync all tables

Format

```
Resync All Tables
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Planned but not yet implemented.

Resync Table

Purpose

Resync a table

Format

```
Resync Table
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Set Automatic Full Sync Frequency

Purpose

Set the frequency of automatic full syncs

Format

```
Set Automatic Full Sync Frequency [ Frequency: <value> ]
```

Options

- The number of sync intervals before a full sync

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Sets the number of normal sync intervals needed before a full sync (i.e. one including deletion checks) should occur. The frequency is the number of sync intervals. If the app was compiled without sync support, then `Get(LastError)` returns 65536 (Sync not supported).

Example 1

Sync every 5 minutes but only check for deletions once per hour

```
Set Automatic Sync Interval [ Interval: 300 ]
Set Automatic Full Sync Frequency [ Frequency: 12 ]
Enable Automatic Sync
```

Related topics

Enable Automatic Sync script step Set Automatic Sync Interval script step

Set Automatic Full Sync On Local Commit

Purpose

Toggle whether automatic sync is full or not after local commit

Format

```
Set Automatic Full Sync On Local Commit [ Enabled: <value> ]
```

Options

- Whether automatic sync on local commit is full or not

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

returns 65536 (Sync not supported).

Example 1

Sync created record to server and check for any remote deletions

```
Enable Automatic Sync
Set Automatic Full Sync On Local Commit [ Enabled: True ]
New Record/Request
Commit Records/Requests [ With dialog: Off ]
```

Related topics

Enable Automatic Sync script step

Set Automatic Sync Enabled

Purpose

Toggle whether automatic sync is enabled or not.

Format

```
Set Automatic Sync Enabled [ Enabled: <value> ]
```

Options

- Whether automatic sync is enabled or not

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Toggles whether the automatic syncing mechanism is enabled or not. When automatic sync is enabled, sync is triggered by local commits and also periodically according to the 'automatic sync interval' property. When automatic sync is disabled, no automatic syncing occurs at all - even if local changes are made. Sync is only triggered by the `Request Sync` and `Sync` script steps. If the app was compiled without sync support then `Get(LastError)` returns 65536 (Sync not supported).

Example 1

Make local commits without syncing

```
Set Automatic Sync Enabled [ Enabled: False ]
New Record/Request
Commit Records/Requests [ With dialog: Off ]
```

Related topics

GetAutomaticSyncEnabled function

Set Automatic Sync Interval

Purpose

Set the interval between automatic syncs

Format

```
Set Automatic Sync Interval [ Interval: <value> ]
```

Options

- The number of seconds between automatic syncs

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Sets the interval between checks for updates on the server, when there have been no local changes. The interval is in seconds. If the interval is 0 then sync will only happen when a local commit needs to be pushed. If the app was compiled without sync support, then `Get(LastError)` returns 65536 (Sync not supported).

Example 1

Sync every minute

```
Set Automatic Sync Interval [ Interval: 60 ]
Enable Automatic Sync
```

Related topics

Enable Automatic Sync script step

Set Deferred Container Cache Limit

Purpose

Set the deferred container cache limit

Format

```
Set Deferred Container Cache Limit [ Number of bytes: <value> ]
```

Options

- The new size of the cache limit

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 2.0

Description

When deferred containers are resolved, they are cached. Use this script step to set the current cache limit, in bytes.

Related topics

GetDeferredContainerCacheSize function GetDeferredContainerCacheLimit function

Set Sync Completion Timeout

Purpose

Set the sync completion timeout

Format

```
Set Sync Completion Timeout [ Timeout: <value> ]
```

Options

- The new sync completion timeout in seconds.

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 2.0

Description

Use this script step to set the amount of time to try and complete sync before aborting. A completion timeout of 0 means do not time out. When the total sync time exceeds the timeout, the sync-timed-out (65539) error is thrown. If a non-zero sync completion timeout has been set, check the error using `Get(LastError)` after using the `Sync` script step to see if sync timed out.

Related topics

GetSyncCompletionTimeout function

Set Sync Solution Account

Purpose

Set the FileMaker Sync solution account

Format

```
Set Sync Solution Account
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Sets the FileMaker Sync solution account username and password to the provided values. Use this script step if you need to change the acces details for a sync solution, such as if you have changed the sync solution name using the `Set Sync Solution Name` script step. Throws an error if any records are locked, else returns 0 (no error).

Example 1

Change the solution name to the another solution, set the appropriate account, and sync data.

```
Set Sync Solution Name [ Address: gSecondSolutionName ]


Set Sync Solution Account [ Account Name: gSecondUserName ; Password: ******** ]


Sync [ Check For Deletions: 1; Timeout:  ]
```

Related topics

Set Sync Solution Name script step GetSyncSolutionAccount function

Set Sync Solution Name

Purpose

Set the FileMaker Sync solution name

Format

`Set Sync Solution Name`

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Sets the FileMaker Sync solution name to the provided value. Use this script step if you need to build a single client app that can sync with multiple different interface files hosted on the same server, or in conjunction with steps such as `Set Server Address` to sync with differently-named interface files on different servers. All local data in syncing tables will be cleared to ensure the correct data is downloaded on next sync. Throws an error if any records are locked, else returns 0 (no error).

Example 1

Validate user in a central server solution, fetch solution name, change the solution name to the user's solution and sync data.

```
Re-Login [ Account Name: gUserName ; Password: ******** ; With dialog: Off ]
If [ Get(LastError) ≠ 0 ]
        Exit Script [ Text Result: "failed to login" ]
End If



Perform Script on Server [ Specified: From list ; "Fetch User Solution" ; Parameter:   ; Wait for
completion: On ]
Set Variable [ $solution_name ; Value: Get(ScriptResult) ]


Set Sync Solution Name [ Address: $solution_name ]


Sync [ Check For Deletions: 1; Timeout:  ]
```

Set Table Sync Constraint

Purpose

Set the sync constraint applied to a table

Format

```
Set Table Sync Constraint
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Set a constraint on a table which is applied to the SQL used to fetch remote records to consider when the table is synced. A sync constraint is a SQL expression which can be concatenated as an 'AND' with any other SQL 'WHERE' clause. It must only reference fields in the table concerned, and no variable interpolation is done. i.e. The script step must be re-executed with an updated constraint if, say, it depends on the current user, and the current user changes. If the app was compiled without sync support, then `Get(LastError)` returns 65536 (Sync not supported). type: calc label: Table datatype: Text showinline: true description: | The name of the table type: calc label: Constraint showinline: tru description: | The constraint to apply.

Example 1

Only sync user-relevant data in table 'Data' when user logs in

```
Loop
    ... wait for Login to be pressed ...

    ... authenticate and exit loop if successful ...
End Loop

Set Table Sync Constraint [ Table: "Data"; Constraint: "userId = " & $$USERID ]

Enable Automatic Sync
```

Related topics

GetTableSyncConstraint function

Set Table Sync Mode

Purpose

Set the sync mode of a table

Format

```
Set Table Sync Mode
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Show Journaled Only

Purpose

Show only journaled records

Format

```
Show Journaled Only
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Planned but not yet implemented.

Sync

Purpose

Sync immediately

See also

How to control how a table syncs

Format

```
Sync [ Check For Deletions: <value> ; Timeout: <value> ]
```

Options

- Whether to check for remote deletions when syncing
- The number of seconds to attempt connection before timeout

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Runs a sync immediately, with optional deletion checks with the given timeout.

The timeout is the time to wait to try and contact the server and start a sync operation, not a timeout for the sync operation as a whole. If a sync does not start within the timeout period then the previous state of a sync request is unaffected - i.e. if 'Sync' was called when there was a pending sync request, and no sync occurs, the sync request will still be present; if 'sync' was called when there was no pending sync request, and no sync occurs, then there will still be no pending sync request.

In order to sync, there must be no locked records in any window. If this is not the case then the operation will have no effect and `Get(LastError)` returns 301 (record locked).

If the sync operation succeeds, `Get(LastError)` returns 0. If the connction times out, it returns 1629 (Connection timed out). If the app was compiled without sync support, then it returns 65536 (Sync not supported).

Otherwise, `Get(LastError)` returns one of the following: 1631: Connection failed 212: Invalid user account and/or password 802: Unable to open file 65539: Sync timed out 65537: Error occurred during sync In this case GetLastExternalErrorDetail() returns a more specific error message.

If an error is generated during sync after a successful login (indicated by error 65537) then no modifications are made to local records, althoug some changes may have been synced to the remote server.

Example 1

Sync and check for deletions immediately, waiting at most 5 seconds to make a connection to FileMaker Server.

```
Sync [ Check For Deletions: True; Timeout: 5 ]
```

Related topics

Set Table Sync Constraint script step

Licensing controls

The LiveCode for FM plugin provides script steps to manage your app's LiveCode for FM license.

Named User Licenses

Apps with a named user license require a user name to be registered. Use the `Register Named User` script step to register a user name fo the license. If, after a period of grace, a user name has not been registered for the license, apps with a named user license will cease to functio

```
Register Named User [ Name: <name> ]
```

Any currently registered named user can be fetched using the `Get Registered Named User` function. This will be empty if no user name has been registered or if the app's license is not a named user license:

```
GetRegisteredNamedUser()
```

Functions

GetRegisteredNamedUser

Purpose

Get named user registered for license

Format

```
GetRegisteredNamedUser
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Returns the name of the user currently registered for the license. If no user has been registered or the license is not a named user license, empty will be returned.

Example 1

Put the currently registered named user into the variable `$NamedUser`

```
Set Variable [ $NamedUser ; Value: GetRegisteredNamedUser() ]
```

Related topics

Register Named User script step

Script Steps

Register Named User

Purpose

Register a named user

Format

```
Register Named User [ Name: <value> ]
```

Options

- The name of the user to be registered

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM 1.5

Description

Register a named user for the license.

Apps with a named user license require a user name to be registered. Use the `Register Named User` script step to register a user name fo the license. If, after a period of grace, a user name has not been registered for the license, apps with a named user license will cease to functio

If the register operation is successful, `Get(LastError)` returns 0. If the named user passed is empty, then error code 5 (invalid command) is returned. If the app's license is not a named user license, then error code 3 (unavailable command) is returned.

Example 1

Register a named user with the content of field `UserName`

```
Register Named User [ Name:  GetField(UserName) ]
```

Related topics

Get Registered Named User function

Location

The LCFM Native plugin provides script steps and functions for control of location tracking on a mobile device for use in an LCFM app.

Location tracking hardware is on when any one of the following is true:

1. An OnLocationChanged script is installed on any window.
2. The `Start Tracking Location` script step has been called in any window without a matching `Stop Tracking Location` scrip step call.
3. Either the `Location`, `LocationValues` or `GetSensor` with `location` or `locationValues` attribute functions have been called until the script execution has stopped or paused.

Functions

GetCurrentLocation

Purpose

Returns the current location's latitude, longitude and accuracy

Format

```
GetCurrentLocation
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes

Where the script step runs	Supported
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns the most recent available location data in the same format as the `Location` function, if no data is available or location tracking is not enabled then empty is returned.

Related topics

Install OnLocationChanged Script script step Start Tracking Location script step

GetCurrentLocationAltitude

Purpose

Returns the current location's altitude

Format

```
GetCurrentLocationAltitude
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns the `altitude` measured in meters of the most recent available location data, if no data is available or location tracking is not enabled then empty is returned.

Related topics

Install OnLocationChanged Script script step Start Tracking Location script step

GetCurrentLocationHorizontalAccuracy

Purpose

Returns the current location's horizontal accuracy

Format

```
GetCurrentLocationHorizontalAccuracy
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes

Where the script step runs	Supported
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns the `horizontal accuracy` measured in meters of the most recent available location data, if no data is available or location tracking is not enabled then empty is returned.

Related topics

Install OnLocationChanged Script script step Start Tracking Location script step

GetCurrentLocationLatitude

Purpose

Returns the current location's latitude

Format

```
GetCurrentLocationLatitude
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns the `latitude` of the most recent available location data, if no data is available or location tracking is not enabled then empty is returned.

Related topics

Install OnLocationChanged Script script step Start Tracking Location script step

GetCurrentLocationLongitude

Purpose

Returns the current location's longitude

Format

```
GetCurrentLocationLongitude
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes

Where the script step runs	Supported
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns the `longitude` of the most recent available location data, if no data is available or location tracking is not enabled then empty is returned.

Related topics

Install OnLocationChanged Script script step Start Tracking Location script step

GetCurrentLocationValues

Purpose

Returns the current location's latitude, longitude, altitude and accuracy aswell as the minutes since the values were returned

Format

```
GetCurrentLocationValues
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns the most recent available location data in the same format as the `LocationValues` function, if no data is available or location tracki is not enabled then empty is returned.

Related topics

Install OnLocationChanged Script script step Start Tracking Location script step

GetCurrentLocationVerticalAccuracy

Purpose

Returns the current location's vertical accuracy

Format

```
GetCurrentLocationVerticalAccuracy
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes

Where the script step runs	Supported
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns the `vertical accuracy` measured in meters of the most recent available location data, if no data is available or location tracking is not enabled then empty is returned.

Related topics

Install OnLocationChanged Script script step Start Tracking Location script step

GetLocationTrackingCurrentAccuracy

Purpose

Returns the location tracking's set accuracy

Format

```
GetLocationTrackingCurrentAccuracy
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns the location tracking's accuracy measured in meters if location tracking is turned on, otherwise it returns empty.

Related topics

Install OnLocationChanged Script script step Start Tracking Location script step

GetLocationTrackingCurrentlyEnabled

Purpose

Returns whether the device's location is currently being tracked

Format

```
GetLocationTrackingCurrentlyEnabled
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes

Where the script step runs	Supported
FileMaker	No

Originated in

LiveCode for FM

Description

Returns `True` if location tracking is turned on, otherwise returns `False`.

Related topics

Install OnLocationChanged Script script step Start Tracking Location script step

GetLocationTrackingError

Purpose

Returns the location tracking error

Format

```
GetLocationTrackingError
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns the location tracking error if an error has occured, otherwise it returns empty.

Related topics

Install OnLocationChanged Script script step Start Tracking Location script step

Script Steps

Install On Location Changed Script

Purpose

Runs a specified script in the current window when the location changes

Format

```
Install On Location Changed Script
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes

Where the script step runs	Supported
FileMaker	No

Originated in

LiveCode for FM

Description

Installs the script specified by file and script name into the active window. If no file name is specified then the active file is used. If no script nam is specified or the file or script specified is invalid then any installed OnLocationChanged script is uninstalled. When an OnLocationChanged script is installed the window is set to track location data behaving in the same way as the `Start Tracking Location` script step and similarly when an OnLocationChanged script is uninstalled the window is set to stop tracking location data behaving in the same way as the `Stop Tracking Location` script step. When the location data changes or an error with location tracking occurs, the OnLocationChanged script step is scheduled to execute the next time a script is not executing and scripts are not paused. This script step is only available on mobile and will return error code `3` if performed on a non-mobile environment or the mobile device's location tracking hardware is not available. If the file name specified is not the name of a valid file then error code `100` will be returned. If the script name specified is not the name of a valid script belonging to the file then error code `104` will be returned. Error codes can be captured by the `Get(LastError)` function.

Example 1

Install script with name "Script" from file with name "File" into the active window

```
Install OnLocationChanged Script [ File: "File" ; Script: "Script" ; Accuracy: 1000 ]
```

Related topics

Install OnLocationChanged Script script step

Start Tracking Location

Purpose

Starts tracking the device location in the current window

Format

```
Start Tracking Location
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Sets the active window to track location data and turns the location tracking hardware on if it isn't already. The accuracy given is measured in meters and is used to configure the location tracking hardware, the location data is not guaranteed to satisfy this accuracy. Multiple windows ca be set to track location data and the accuracy set is the lowest/best across all windows. When location tracking is turned on with this script step no guarantee is made when location data will be made available and location data is only updated when no script is executing or is paused. If a window set to track location is closed then `Stop Tracking Location` is implicitly called. If an error occurs with location tracking then all windows are set to stop tracking, location tracking stops and the error can be fetched using the `GetLocationTrackingError` function, this error is cleared when location tracking starts again. This script step is only available on mobile and will return error code `3` which can be captured by the `Get(LastError)` function if performed on a non-mobile environment or the mobile device's location tracking hardware is not available.

Example 1

Start the device location tracking with accuracy set to 1000

```
Start Tracking Location [ Accuracy: 1000 ]
```

Related topics

Stop Tracking Location script step

Stop Tracking Location

Purpose

Stops tracking the device location in the current window

Format

```
Stop Tracking Location
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Sets the active window to stop tracking location data and attempts to turn the location tracking hardware off. The location tracking hardware wil be turned off the next time no script is executing or is paused, this will only occur if no windows are set to track location data and no windows have an installed OnLocationChanged script. When the location tracking hardware is turned off any cached location data is cleared. This script step is only available on mobile and will return error code `3` which can be captured by the `Get(LastError)` function if performed on a non-mobile environment or the mobile device's location tracking hardware is not available.

Example 1

Stop the device location tracking

```
Stop Tracking Location
```

Related topics

Start Tracking Location script step

Miscellaneous

The LiveCode for FM plugin provides the following miscellaneous set of useful script steps and functions for use in an app.

Functions

GetAllowDeviceSleep

Purpose

Determine whether the device is allowed to sleep

Format

```
GetAllowDeviceSleep
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns `False` if the device was set to prevent sleeping with `Prevent Device Sleep` otherwise returns `True`.

Related topics

Set Allow Device Sleep script step

GetAllowOrientationChanges

Purpose

Determine whether the device is allowed to make orientation changes

Format

```
GetAllowOrientationChanges
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns `True` if the device is allowed to make orientation changes, otherwise returns `False`.

Related topics

Set Allow Orientation Changes script step

GetServerAddress

Purpose

Returns the FileMaker server address

Format

```
GetServerAddress
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Returns the FileMaker server address.

Script Steps

Allow Device Sleep

Purpose

Allow device to sleep

Format

```
Allow Device Sleep
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Allows the device to sleep when idle. By default an app allows the device to sleep, this script step is provided to resume this behaviour after `Prevent Device Sleep` is performed. This script step is only available on mobile and will return error code `3` which can be captured by th `Get(LastError)` function if performed on a non-mobile environment.

Example 1

Allow the device to sleep

```
Allow Device Sleep
```

Related topics

Allow Device Sleep script step

Allow Orientation Changes

Purpose

Allow device to change the orientation of the app

Format

```
Allow Orientation Changes
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Allows the device to change the orientation of the application. By default an app allows orientation changes, this script step is provided to resume this behaviour after `Prevent Orientation Changes` is performed. This script step is only available on mobile and will return error code `3` which can be captured by the `Get(LastError)` function if performed on a non-mobile environment.

Example 1

Allow the device to change the orientation

```
Allow Orientation Changes
```

Related topics

Allow Orientation Changes script step

Prevent Device Sleep

Purpose

Prevent device from sleeping

Format

```
Prevent Device Sleep
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Prevents the device from sleeping when idle. This script step is only available on mobile and will return error code `3` which can be captured b the `Get(LastError)` function if performed on a non-mobile environment.

Example 1

Prevent the device from sleeping

```
Prevent Device Sleep
```

Related topics

Prevent Device Sleep script step

Prevent Orientation Changes

Purpose

Prevent device from changing the orientation of the app

Format

```
Prevent Orientation Changes
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Prevents the device from changing the orientation of the application. This script step is only available on mobile and will return error code `3` which can be captured by the `Get(LastError)` function if performed on a non-mobile environment.

Example 1

Prevent the device from changing the orientation

```
Prevent Orientation Changes
```

Related topics

Prevent Orientation Changes script step

Set Allow Device Sleep

Purpose

Allows/Prevents the device from sleeping

Format

```
Set Allow Device Sleep
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Allows or Prevents the device from sleeping when idle depending on the result of the `Allow` calculation. If the calculation evaluates to `True` then the device will be allowed to sleep; if the calculation evaluates to `False` then the device will be prevented from sleeping. This script step i only available on mobile and will return error code `3` which can be captured by the `Get(LastError)` function if performed on a non-mobile environment.

Example 1

Allow the device to sleep

```
Set Allow Device Sleep [ Allow: True ]
```

Related topics

Set Allow Device Sleep script step

Set Allow Orientation Changes

Purpose

Allows/Prevents the device from changing the orientation of the app

Format

```
Set Allow Orientation Changes
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Allows or Prevents the device from changing the orientation of the application depending on the result of the `Allow` calculation. If the calculation evaluates to `True` then the device will be allowed to change the app orientation; if the calculation evaluates to `False` then the device will be prevented from changing the app orientation. This script step is only available on mobile and will return error code `3` which can b captured by the `Get(LastError)` function if performed on a non-mobile environment.

Example 1

Allow the device to change the orientation

```
Set Allow Orientation Changes [ Allow: True ]
```

Related topics

Set Allow Orientation Changes script step

Set Server Address

Purpose

Set the FileMaker server address

Format

```
Set Server Address
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Sets the FileMaker server address to the provided value. Use this script step if you need to build a single client app that can sync with interface files hosted on different servers, or that can perform remote operations like `Perform Script On Server`, or `Re-Login` or other account-related functionality in solutions hosted on different servers. If sync is enabled, then all local data in syncing tables will be cleared to ensure the correct data is downloaded on next sync. Throws an error if any records are locked, else returns 0 (no error).

Example 1

Validate user in a central server solution, fetch server address, change the server address to the user's server and sync data.

```
Re-Login [ Account Name: gUserName ; Password: ******** ; With dialog: Off ]
If [ Get(LastError) ≠ 0 ]
        Exit Script [ Text Result: "failed to login" ]
End If


Perform Script on Server [ Specified: From list ; "Fetch User Server Address" ; Parameter:   ; Wait for
completion: On ]
Set Variable [ $server_address ; Value: Get(ScriptResult) ]


Set Server Address [ Address: $server_address ]


Perform Script on Server [ Specified: From list ; "Fetch User Id" ; Parameter:   ; Wait for completion:
On ]
Set Variable [ $user_id ; Value: Get(ScriptResult) ]


Set Table Sync Constraint [ Table: "UserData" ; Constraint: "\"fkUserId\"='' & $user_id & '" ]
Sync [ Check For Deletions: 1; Timeout:   ]
```

Snapshot

The LiveCode for FM plugin provides script steps allowing apps to take snapshots of windows. The snapshots taken will be compressed PNGs and can be entered into a variable or field.

Script Steps

Take Snapshot Of Current Window Into Variable

Purpose

Take a snapshot of the current window

Format

```
Take Snapshot Of Current Window Into Variable
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Takes a snapshot of the current window and enters it into a variable or field as a compressed PNG. You must specify a target variable, specify target field on the current layout, click in a field, or use the `Go to Field` script step before performing this script step. If Target is not specifie the data is placed in the active field. Otherwise, this script step returns an error code that can be captured with the `Get(LastError)` function If specified, the target field must be a container field. Otherwise, this script step returns an error code that can be captured with the `Get(LastError)` function.

Example 1

Enter a snapshot of the current window into the variable `$Snapshot`

```
Take Snapshot Of Current Window [ Select ; Target: $Snapshot ]
```

Related topics

Take Snapshot Of Window script step

Take Snapshot Of Window Into Variable

Purpose

Format

```
Take Snapshot Of Window Into Variable
```

Compatibility

Where the script step runs	Supported
LiveCode for FM - Android	Yes
LiveCode for FM - iOS	Yes
LiveCode for FM - macOS	Yes
FileMaker	No

Originated in

LiveCode for FM

Description

Sync controls

The LiveCode for FM plugin provides the ability to control the application's sync settings within FileMaker solution scripts, as well as the ability actually trigger syncs manually.

If automatic sync is enabled, the application will sync with the server after every local commit, and also periodically when idle (at a frequency governed by the automatic sync interval). If automatic sync is disabled, sync can only be triggered explicitly through the `Sync` or `Request Sync` script steps.