## NAME

Image::ExifTool − Read and write meta information

## SYNOPSIS

```
use Image::ExifTool qw(:Public);

# ---- Simple procedural usage ----

# Get hash of meta information tag names/values from an image
$info = ImageInfo('a.jpg');

# ---- Object-oriented usage ----

# Create a new Image::ExifTool object
$exifTool = Image::ExifTool->new;

# Extract meta information from an image
$exifTool->ExtractInfo($file, \%options);

# Get list of tags in the order they were found in the file
@tagList = $exifTool->GetFoundTags('File');

# Get the value of a specified tag
$value = $exifTool->GetValue($tag, $type);

# Get a tag description
$description = $exifTool->GetDescription($tag);

# Get the group name associated with this tag
$group = $exifTool->GetGroup($tag, $family);

# Set a new value for a tag
$exifTool->SetNewValue($tag, $newValue);

# Write new meta information to a file
$success = $exifTool->WriteInfo($srcfile, $dstfile);

# ...plus a host of other useful methods...
```

## DESCRIPTION

Reads and writes meta information in a wide variety of files, including the maker notes of many digital cameras by various manufacturers such as Apple, Canon, Casio, DJI, FLIR, FujiFilm, GE, Google, GoPro, HP, JVC/Victor, Kodak, Leaf, Minolta/Konica−Minolta, Nikon, Nintendo, Olympus/Epson, Panasonic/Leica, Pentax/Asahi, Phase One, Reconyx, Ricoh, Samsung, Sanyo, Sigma/Foveon and Sony.

Below is a list of file types and meta information formats currently supported by ExifTool (r = read, w = write, c = create):

File Types

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 360 | r/w | DR4 | r/w/c | JNG | r/w | ODP | r | RTF | r |
| 3FR | r | DSF | r | JP2 | r/w | ODS | r | RW2 | r/w |
| 3G2 | r/w | DSS | r | JPEG | r/w | ODT | r | RWL | r/w |
| 3GP | r/w | DV | r | JSON | r | OFR | r | RWZ | r |
| 7Z | r | DVB | r/w | JXL | r/w | OGG | r | RM | r |
| A | r | DVR-MS | r | K25 | r | OGV | r | SEQ | r |
| AA | r | DYLIB | r | KDC | r | ONP | r | SKETCH | r |
| AAC | r | EIP | r | KEY | r | OPUS | r | SO | r |
| AAE | r | EPS | r/w | KVAR | r | ORF | r/w | SR2 | r/w |
| AAX | r/w | EPUB | r | LA | r | ORI | r/w | SRF | r |
| ACR | r | ERF | r/w | LFP | r | OTF | r | SRW | r/w |
| AFM | r | EXE | r | LIF | r | PAC | r | SVG | r |
| AI | r/w | EXIF | r/w/c | LNK | r | PAGES | r | SWF | r |
| AIFF | r | EXR | r | LRV | r/w | PBM | r/w | THM | r/w |
| APE | r | EXV | r/w/c | M2TS | r | PCAP | r | TIFF | r/w |
| ARQ | r/w | F4A/V | r/w | M4A/V | r/w | PCAPNG | r | TNEF | r |
| ARW | r/w | FFF | r/w | MACOS | r | PCD | r | TORRENT | r |
| ASF | r | FIT | r | MAX | r | PCX | r | TTC | r |
| AVI | r | FITS | r | MEF | r/w | PDB | r | TTF | r |
| AVIF | r/w | FLA | r | MIE | r/w/c | PDF | r/w | TXT | r |
| AZW | r | FLAC | r | MIFF | r | PEF | r/w | URL | r |
| BMP | r | FLIF | r/w | MKA | r | PFA | r | VCF | r |
| BPG | r | FLV | r | MKS | r | PFB | r | VNT | r |
| BTF | r | FPF | r | MKV | r | PFM | r | VRD | r/w/c |
| C2PA | r | FPX | r | MNG | r/w | PGF | r | VSD | r |
| CHM | r | GIF | r/w | MOBI | r | PGM | r/w | VSDX | r |
| COS | r | GLV | r/w | MODD | r | PLIST | r | WAV | r |
| CR2 | r/w | GPR | r/w | MOI | r | PICT | r | WDP | r/w |
| CR3 | r/w | GZ | r | MOS | r/w | PMP | r | WEBP | r/w |
| CRM | r/w | HDP | r/w | MOV | r/w | PNG | r/w | WEBM | r |
| CRW | r/w | HDR | r | MP3 | r | PPM | r/w | WMA | r |
| CS1 | r/w | HEIC | r/w | MP4 | r/w | PPT | r | WMV | r |
| CSV | r | HEIF | r/w | MPC | r | PPTX | r | WOFF | r |
| CUR | r | HTML | r | MPG | r | PS | r/w | WOFF2 | r |
| CZI | r | ICC | r/w/c | MPO | r/w | PSB | r/w | WPG | r |
| DCM | r | ICO | r | MQV | r/w | PSD | r/w | WTV | r |
| DCP | r/w | ICS | r | MRC | r | PSP | r | WV | r |
| DCR | r | IDML | r | MRW | r/w | QTIF | r/w | X3F | r/w |
| DFONT | r | IIQ | r/w | MXF | r | R3D | r | XCF | r |
| DIVX | r | IND | r/w | NEF | r/w | RA | r | XISF | r |
| DJVU | r | INSP | r/w | NKA | r | RAF | r/w | XLS | r |
| DLL | r | INSV | r | NKSC | r/w | RAM | r | XLSX | r |
| DNG | r/w | INX | r | NRW | r/w | RAR | r | XMP | r/w/c |
| DOC | r | ISO | r | NUMBERS | r | RAW | r/w | ZIP | r |
| DOCX | r | ITC | r | NXD | r | RIFF | r | | |
| DPX | r | J2C | r | O | r | RSRC | r | | |

Meta Information

| | | | | | |
|---|---|---|---|---|---|
| EXIF | r/w/c | CIFF | r/w | Ricoh RMETA | r |
| GPS | r/w/c | AFCP | r/w | Picture Info | r |
| IPTC | r/w/c | Kodak Meta | r/w | Adobe APP14 | r |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| XMP | r/w/c | | FotoStation | r/w | | MPF | r |
| MakerNotes | r/w/c | | PhotoMechanic | r/w | | Stim | r |
| Photoshop IRB | r/w/c | | JPEG 2000 | r | | DPX | r |
| ICC Profile | r/w/c | | DICOM | r | | APE | r |
| MIE | r/w/c | | Flash | r | | Vorbis | r |
| JFIF | r/w/c | | FlashPix | r | | SPIFF | r |
| Ducky APP12 | r/w/c | | QuickTime | r | | DjVu | r |
| PDF | r/w/c | | Matroska | r | | M2TS | r |
| PNG | r/w/c | | MXF | r | | PE/COFF | r |
| Canon VRD | r/w/c | | PrintIM | r | | AVCHD | r |
| Nikon Capture | r/w/c | | FLAC | r | | ZIP | r |
| GeoTIFF | r/w/c | | ID3 | r | | (and more) | |

## CONFIGURATION

User-defined tags can be added via the ExifTool configuration file, or by defining the `%Image::ExifTool::UserDefined` hash before calling any ExifTool methods. See ''ExifTool_config'' in the ExifTool distribution for more details.

By default ExifTool looks for a configuration file named ''.ExifTool_config'' first in your home directory, then in the directory of the application script, but a different directory may be specified by setting the EXIFTOOL_HOME environment variable, or a different file may be specified by setting the ExifTool `configFile` variable before using Image::ExifTool. For example:

```
BEGIN { $Image::ExifTool::configFile = '/Users/phil/myconfig.cfg' }
use Image::ExifTool;
```

The configuration feature may also be disabled by setting `configFile` to an empty string:

```
BEGIN { $Image::ExifTool::configFile = '' }
use Image::ExifTool;
```

## EXPORTS

Exports nothing by default, but ''ImageInfo'' and all static methods may be exported with the `:Public` export list.

## METHODS

All ExifTool features are accessed through the methods of the public interface listed below. Other Image::ExifTool methods and modules should not be accessed directly because their interface may change with future versions.

None of these methods should ever die or issue warnings to STDERR if called with the proper arguments (with the exception of ''SetNewValue'' which may send an error message to STDERR, but only when called in scalar context). Error and warning messages that occur during processing are stored in the values of the Error and Warning tags, and are accessible via the ''GetValue'' method to retrieve a single Error or Warning message, or ''GetInfo'' to retrieve any number of them.

The ExifTool methods are not thread safe.

### new

Creates a new ExifTool object.

```
$exifTool = Image::ExifTool->new;
```

One ExifTool object may be used to process many files, so creating multiple ExifTool objects usually is not necessary.

Note that ExifTool uses AUTOLOAD to load non-member methods, so any class using Image::ExifTool as a base class must define an AUTOLOAD which calls **Image::ExifTool::DoAutoLoad()**. eg)

```
sub AUTOLOAD
{
    Image::ExifTool::DoAutoLoad($AUTOLOAD, @_);
}
```

**ImageInfo**

Read image file and return meta information. This is the one step function for retrieving meta information from an image. Internally, ‘‘ImageInfo’’ calls ‘‘ExtractInfo’’ to extract the information, ‘‘GetInfo’’ to generate the information hash, and ‘‘GetTagList’’ for the returned tag list.

```
# return meta information for 2 tags only (procedural)
$info = ImageInfo($filename, $tag1, $tag2);

# return information about an open image file (object-oriented)
$info = $exifTool->ImageInfo(\*FILE);

# return information from image data in memory for specified tags
%options = (PrintConv => 0);
@tagList = qw(filename imagesize xmp:creator exif:* -ifd1:*);
$info = ImageInfo(\$imageData, \@tagList, \%options);

# extract information from an embedded thumbnail image
$info = ImageInfo('image.jpg', 'thumbnailimage');
$thumbInfo = ImageInfo($$info{ThumbnailImage});
```

Inputs:

‘‘ImageInfo’’ is very flexible about the input arguments, and interprets them based on their type. It may be called with one or more arguments. The one required argument is either a SCALAR (the image file name), a file reference (a reference to the image file) or a SCALAR reference (a reference to the image in memory). Other arguments are optional. The order of the arguments is not significant, except that the first SCALAR is taken to be the file name unless a file reference or scalar reference comes earlier in the argument list.

Below is an explanation of how the ‘‘ImageInfo’’ function arguments are interpreted:

ExifTool ref

‘‘ImageInfo’’ may be called with an ExifTool object if desired. Advantages of using the object-oriented form are that options may be set before calling ‘‘ImageInfo’’, and the object may be used afterward to access member functions. Must be the first argument if used.

SCALAR

The first scalar argument is taken to be the file name unless an earlier argument specified the image data via a file reference (file ref) or data reference (SCALAR ref). The remaining scalar arguments are names of tags for requested information. All tags are returned if no tags are specified.

Tag names are case-insensitive and may be prefixed by optional group names separated by colons. A group name may begin with a family number (eg. ’1IPTC:Keywords’), to restrict matches to a specific family. In the tag name, a ’?’ matches any single character and a ’*’ matches zero or more characters. Thus ’GROUP:*’ represents all tags in a specific group. Wildcards may not be used in group names, with the exception that a group name of ’*’ may be used to extract all available instances of a tag regardless of the ‘‘Duplicates’’ setting (eg. ’*:WhiteBalance’). Multiple groups may be specified (eg. ’EXIF:Time:*’ extracts all EXIF Time tags). And finally, a leading ’-’ indicates a tag to be excluded (eg. ’-IFD1:*’), or a trailing ’#’ causes the ValueConv value to be returned for this tag.

Note that keys in the returned information hash and elements of the returned tag list are not necessarily the same as these tag names because group names are removed, the case may be changed, and an instance number may be added. For this reason it is best to use either the keys of

the returned hash or the elements of the returned tag list when accessing the tag values.

See Image::ExifTool::TagNames for a complete list of ExifTool tag names.

File ref
: A reference to an open image file. If you use this method (or a SCALAR reference) to access information in an image, the FileName and Directory tags will not be returned. (Also, a number of the File System tags will not be returned unless it is a plain file.) Image processing begins at the current file position, and on return the file position is unspecified. May be either a standard filehandle, or a reference to a File::RandomAccess object. Note that the file remains open and must be closed by the caller after ''ImageInfo'' returns.

  [Advanced: To allow a non-rewindable stream (eg. a network socket) to be re-read after processing with ExifTool, first wrap the file reference in a File::RandomAccess object, then pass this object to ''ImageInfo''. The File::RandomAccess object will buffer the file if necessary, and may be used to re-read the file after ''ImageInfo'' returns.]

SCALAR ref
: A reference to image data in memory.

ARRAY ref
: Reference to a list of tag names. On entry, any elements in the list are added to the list of requested tags. Tags with names beginning with '−' are excluded. On return, this list is updated to contain an ordered list of tag keys for the returned information.

  There will be 1:1 correspondence between the requested tags and the returned tag keys only if the ''Duplicates'' option is 0 and ''Sort'' is 'Input'. (With ''Duplicates'' enabled, there may be more entries in the returned list of tag keys, and with other ''Sort'' settings the entries may not be in the same order as requested.) If a requested tag doesn't exist, a tag key is still generated, but the tag value is undefined.

  **Note:** Do not reuse this list in subsequent calls to ''ImageInfo'' because it returns tag keys, not names, and the list will grow for each call resulting in increasingly slower performance.

HASH ref
: Reference to a hash containing the options settings valid for this call only. See ''Options'' documentation below for a list of available options. Options specified as arguments to ''ImageInfo'' take precedence over ''Options'' settings.

Return Values:
: ''ImageInfo'' returns a reference to a hash of tag−key/value pairs. The tag keys are identifiers — essentially case-sensitive tag names with an appended instance number if multiple tags with the same name were extracted from the image. Many of the ExifTool functions require a tag key as an argument. Use ''GetTagName [static]'' to get the tag name for a given tag key. Note that the case of the tag names may not be the same as requested. Here is a simple example to print out the information returned by ''ImageInfo'':

```
foreach (sort keys %$info) {
    print "$_ => $$info{$_}\n";
}
```

  Values of the returned hash are usually simple scalars, but a scalar reference is used to indicate binary data and an array reference may be used to indicate a list. Also, a hash reference may be returned if the ''Struct'' option is used (see the ''OrderedKeys'' option to obtain the hash keys). Lists of values are joined by commas into a single string only if the PrintConv option is enabled and the ListJoin option is enabled (which are the defaults). Note that binary values are not necessarily extracted unless specifically requested, or the Binary option is enabled and the tag is not specifically excluded. If not extracted the value is a reference to a string of the form ''Binary data ##### bytes''.

  The code below gives an example of how to handle these return values, as well as illustrating the use of other ExifTool functions:

```
use Image::ExifTool;
my $exifTool = Image::ExifTool->new;
$exifTool->Options(Unknown => 1);
my $info = $exifTool->ImageInfo('a.jpg');
my $group = '';
my $tag;
foreach $tag ($exifTool->GetFoundTags('Group0')) {
    if ($group ne $exifTool->GetGroup($tag)) {
        $group = $exifTool->GetGroup($tag);
        print "---- $group ----\n";
    }
    my $val = $info->{$tag};
    if (ref $val eq 'SCALAR') {
        if ($$val =~ /^Binary data/) {
            $val = "($$val)";
        } else {
            my $len = length($$val);
            $val = "(Binary data $len bytes)";
        }
    }
    printf("%-32s : %s\n", $exifTool->GetDescription($tag), $val);
}
```

Notes:

ExifTool returns all values as byte strings of encoded characters. Perl wide characters are not used. See ''CHARACTER ENCODINGS'' for details about the encodings. By default, most returned values are encoded in UTF-8. For these, **Encode::decode_utf8()** may be used to convert to a sequence of logical Perl characters.

As well as tags representing information extracted from the image, the following Extra tags generated by ExifTool may be returned:

```
ExifToolVersion - The ExifTool version number.

Error - An error message if the image could not be processed.

Warning - A warning message if problems were encountered while
          processing the image.
```

**Options**

Get/set ExifTool options. This function can be called to set the default options for an ExifTool object. Options set this way are in effect for all function calls but may be overridden by options passed as arguments to some functions. Option names are not case sensitive, but option values are.

The default option values may be changed by defining a `%Image::ExifTool::UserDefined::Options` hash. See the ExifTool_config file in the full ExifTool distribution for examples. Unless otherwise noted, a default of undef has the same effect as a value of 0 for options with numerical values.

```
# exclude the 'OwnerName' tag from returned information
$exifTool->Options(Exclude => 'OwnerName');

# only get information in EXIF or MakerNotes groups
$exifTool->Options(Group0 => ['EXIF', 'MakerNotes']);

# ignore information from IFD1
$exifTool->Options(Group1 => '-IFD1');
```

```
# sort by groups in family 2, and extract unknown tags
$exifTool->Options(Sort => 'Group2', Unknown => 1);

# reset DateFormat option
$exifTool->Options(DateFormat => undef);

# do not extract duplicate tag names
$oldSetting = $exifTool->Options(Duplicates => 0);

# get current Verbose setting
$isVerbose = $exifTool->Options('Verbose');

# set a user parameter
$exifTool->Options(UserParam => 'MyParam=some value');
```

Inputs:

0) ExifTool object reference

1) Option parameter name (case-insensitive)

2) [optional] Option parameter value (may be undef to clear option)

3-N) [optional] Additional parameter/value pairs

Option Parameters:

Note that these API options may also be used in the exiftool application via the command-line **-api** option.

Binary

Flag to extract the value data for all binary tags. Tag values representing large binary data blocks (eg. ThumbnailImage) are not necessarily extracted unless this option is set or the tag is specifically requested by name. Default is undef.

BlockExtract

Flag to extract some directories (mentioned in the ExifTool tag name documentation) as a block. Setting this to a value of 2 also prevents parsing the block to extract tags contained within.

ByteOrder

The byte order for newly created EXIF segments when writing. Note that if EXIF information already exists, the existing order is maintained. Valid values are 'MM', 'II' and undef. If ByteOrder is not defined (the default), then the maker note byte order is used (if they are being copied), otherwise big-endian ('MM') order is assumed. This can also be set via the ExifByteOrder tag, but the ByteOrder option takes precedence if both are set.

ByteUnit

Units for print conversion of FileSize and other byte values. Either 'SI' (eg. kB for 1000 bytes) or 'Binary' (eg. KiB for 1024 bytes). Default is 'SI'.

Charset

Character set for encoding character tag values passed to/from ExifTool with code points above U+007F. Default is 'UTF8'. Valid values are listed below, case is not significant:

| Value | Alias(es) | Description |
|---|---|---|
| UTF8 | cp65001, UTF-8 | UTF-8 characters |
| Latin | cp1252, Latin1 | Windows Latin1 (West European) |
| Latin2 | cp1250 | Windows Latin2 (Central European) |
| Cyrillic | cp1251, Russian | Windows Cyrillic |
| Greek | cp1253 | Windows Greek |
| Turkish | cp1254 | Windows Turkish |
| Hebrew | cp1255 | Windows Hebrew |
| Arabic | cp1256 | Windows Arabic |
| Baltic | cp1257 | Windows Baltic |
| Vietnam | cp1258 | Windows Vietnamese |
| Thai | cp874 | Windows Thai |
| DOSLatinUS | cp437 | DOS Latin US |
| DOSLatin1 | cp850 | DOS Latin1 |
| DOSCyrillic | cp866 | DOS Cyrillic |
| MacRoman | cp10000, Roman | Macintosh Roman |
| MacLatin2 | cp10029 | Macintosh Latin2 (Central Europe) |
| MacCyrillic | cp10007 | Macintosh Cyrillic |
| MacGreek | cp10006 | Macintosh Greek |
| MacTurkish | cp10081 | Macintosh Turkish |
| MacRomanian | cp10010 | Macintosh Romanian |
| MacIceland | cp10079 | Macintosh Icelandic |
| MacCroatian | cp10082 | Macintosh Croatian |

Note that this option affects some types of information when reading/writing the file and other types when getting/setting tag values, so it must be defined for both types of access. See the ''CHARACTER ENCODINGS'' section for more information about the handling of special characters.

CharsetEXIF

Internal encoding to use for stored EXIF ''ASCII'' string values. May also be set to undef to pass through EXIF ''ASCII'' values without recoding. Set to ''UTF8'' to conform with the MWG recommendation. Default is undef.

CharsetFileName

External character set used for file names passed to ExifTool functions. Default is undef but ''UTF8'' is assumed in Windows if the file name contains special characters and is valid UTF8. May also be set to an empty string to avoid ''encoding must be specified'' warnings on Windows.

CharsetID3

Internal encoding to assume for ID3v1 strings. By the specification ID3v1 strings should be encoded in ISO 8859-1 (essentially Latin), but some applications may use local encoding instead. Default is 'Latin'.

CharsetIPTC

Fallback internal IPTC character set to assume if IPTC information contains no CodedCharacterSet tag. Possible values are the same as the ''Charset'' option. Default is 'Latin'.

Note that this option affects some types of information when reading/writing the file and other types when getting/setting tag values, so it must be defined for both types of access.

CharsetPhotoshop

Internal encoding to assume for Photoshop IRB resource names. Default is 'Latin'.

CharsetQuickTime

Internal encoding to assume for QuickTime strings stored with an unspecified encoding. Default is 'MacRoman'.

CharsetRIFF

Internal encoding to assume for strings in RIFF metadata (eg. AVI and WAV files). The default value of 0 assumes ‘‘Latin’’ encoding unless otherwise specified by the RIFF CSET chunk. Set to undef to pass through strings without recoding. Default is 0.

Compact

Comma-delimited list of settings for writing compact XMP. Below is a list of available settings. Note that ’NoPadding’ effects only embedded XMP since padding is never written for stand-alone XMP files. Also note that ’OneDesc’ is not recommended when writing XMP larger than 64 KiB to a JPG file because it interferes with ExifTool’s technique of splitting off large rdf:Description elements into the extended XMP. Case is not significant for any of these options. Aliases are given in brackets. Default is undef.

```
NoPadding − Avoid 2 KiB of recommended padding at end of XMP (NoPad)
NoIndent  − No spaces to indent lines (NoSpace, NoSpaces)
NoNewline − Avoid unnecessary newlines (NoNewlines)
Shorthand − Use XMP Shorthand format
OneDesc   − Combine properties into a single rdf:Description (OneDescr)
AllSpace  − Equivalent to 'NoPadding,NoIndent,NoNewline'
AllFormat − Equivalent to 'Shorthand,OneDesc'
All       − Equivalent to 'AllSpace,AllFormat'
```

Composite

Flag to generate Composite tags when extracting information. Default is 1.

Compress

Flag to write new values in compressed format if possible. Has no effect unless the relevant compression library is available. Valid when writing metadata to PNG, JXL, HEIC or MIE images. Setting this to zero causes JXL metadata to be rewritten as uncompressed when edited. Default is undef.

CoordFormat

Format for printing GPS coordinates. This is a printf format string with specifiers for degrees, minutes and seconds in that order, however minutes and seconds may be omitted. If the hemisphere is known, a reference direction (N, S, E or W) is appended to each printed coordinate, but adding a + or − to the first format specifier (eg. `%+.6f` or `%−.6f`) prints a signed coordinate instead (+ also adds a leading plus sign to positive coordinates, while − does not). For example, the following table gives the output for the same coordinate using various formats:

```
    CoordFormat        Example Output
------------------  ------------------
q{%d deg %d' %.2f"}  54 deg 59' 22.80"  (default for reading)
q{%d %d %.8f}        54 59 22.80000000  (default for copying)
q{%d deg %.4f min}   54 deg 59.3800 min
q{%.6f degrees}      54.989667 degrees
```

Note: To avoid loss of precision, the default coordinate format is different when copying tags with ‘‘SetNewValuesFromFile’’.

DateFormat

Format for printing date/time values. See`strftime` in the POSIX package and <https://exiftool.org/filename.html#codes> for details about the format string. If the date can not be converted, the value is left unchanged unless the StrictDate option is set. ExifTool adds a few additional format specifiers (`%f`, `%s` and `%z`), see <https://exiftool.org/filename.html#codes> for more details. The inverse conversion (ie. when calling ‘‘SetNewValue’’) is performed only if POSIX::strptime or Time::Piece is installed. The default setting of undef causes date/time values to remain in standard EXIF format (similar to a DateFormat of ‘‘%Y:%m:%d `%H:`%M:%S’’).

Duplicates

Flag to return values from tags with duplicate names when extracting information. Default is 1. Forced to 1 when copying tags with ''SetNewValuesFromFile''.

Escape

Escape special characters in extracted values for HTML or XML. Also unescapes HTML or XML character entities in input values passed to ''SetNewValue''. Valid settings are 'HTML', 'XML' or undef. Default is undef.

Exclude

Exclude specified tags when extracting information. Note that this option is applied after all of the tags have already been loaded into memory (so different tags may be excluded in subsequent calls to ''GetInfo''). See the IgnoreTags option to save memory by not loading the tags in the first place. The option value is either a tag name or reference to a list of tag names to exclude. The case of tag names is not significant. This option is ignored for specifically requested tags. Tags may also be excluded by preceding their name with a '−' in the arguments to ''ImageInfo''.

ExtendedXMP

This setting affects the reading and editing of extended XMP in JPEG images. According to the XMP specification, extended XMP is only valid if it has the GUID specified by the HasExtendedXMP tag, so by default ExifTool will ignore other extended XMP, but this option allows full control over the extended XMP to be extracted.

```
    0   − Ignore all extended XMP
    1   − Read extended XMP with valid GUID only (default)
    2   − Read extended XMP with any GUID
<guid> − Read extended XMP with a specific GUID
```

ExtractEmbedded

Flag to extract information from embedded documents in EPS files, embedded EPS information and JPEG and Jpeg2000 images in PDF files, embedded MPF images in JPEG and MPO files, metadata after the first Cluster in MKV files, timed metadata in videos, all frames of a multipart EXR image, and the resource fork of Mac OS files. A setting of 2 also causes the H264 video stream in MP4 files to be parsed until the first SEI message is decoded, or 3 to parse the entire H264 stream in MP4 videos and the entire M2TS file to look for any unlisted program containing GPS metadata. Default is undef.

FastScan

Flag to increase speed when reading files by avoiding extraction of some types of metadata. With this option set to 1, ExifTool will not scan to the end of a JPEG image to check for an AFCP, CanonVRD, FotoStation, PhotoMechanic, MIE or PreviewImage trailer. This also stops the parsing after the first comment in GIF images, and at the audio/video data of RIFF-format files (AVI, WAV, etc), so any trailing metadata (eg. XMP written by some utilities) may be missed. Also disables input buffering for some types of files to reduce memory usage when reading from a non-seekable stream, and bypasses CRC validation for speed when writing PNG files. When combined with the ScanForXMP option, prevents scanning for XMP in recognized file types. With a value of 2, ExifTool will also avoid extracting any EXIF MakerNote information, and will stop processing at the IDAT chunk of PNG images and the mdat atom in QuickTime-format files. (By the PNG specification, metadata is allowed after IDAT, but ExifTool always writes it before because some utilities will ignore it otherwise.) When set to 3 or higher, only pseudo system tags and FileType are generated. For 3, the file header is read to provide an educated guess at FileType. For 4, the file is not read at all and FileType is determined based on the file's extension. For 5, generation of Composite tags is also disabled (like setting ''Composite'' to 0). Default is undef.

Filter

Perl expression used to filter values for all tags. The expression acts on the value of the Perl default variable ($_), and changes the value of this variable as required. The current ExifTool object may be accessed through `$self`. The value is not changed if `$_` is set to undef. List

items are filtered individually. Applies to all returned values unless PrintConv option is disabled.

FilterW

Perl expression used to filter PrintConv values when writing. The expression acts on the value of the Perl default variable ($_), and changes the value of this variable as required. The current ExifTool object may be accessed through `$self`. The tag is not written if$_ is set to undef.

FixBase

Fix maker notes base offset. A common problem with image editing software is that offsets in the maker notes are not adjusted properly when the file is modified. This may cause the wrong values to be extracted for some maker note entries when reading the edited file. FixBase specifies an integer value to be added to the maker notes base offset. It may also be set to the empty string ('') for ExifTool will take its best guess at the correct base, or undef (the default) for no base adjustment.

GeoDir

[Not a real option] Provided as a convenience to allow `$Image::ExifTool::geoDir` to be set at runtime. This variable specifies the directory for the Geolocation databases, and is used only once when these databases are loaded.

Geolocation

Flag to generate geolocation tags based on the GPSLatitude/GPSLongitude or City/State/Province/Country read from a file. This feature uses an included database with cities over a population of 2000 from geonames.org. May be set to a string of the form ''Lat,Lon'' (eg. ''44.56,−72.33'') or city with optional state/province, country and/or country code (eg. ''Paris,France'') to act as a default for files not containing GPS or geolocation information, or include tag names with leading dollar signs separated by commas to specify the tags to use for the geolocation input. When ''Lat,Lon'' is specified, ''num=##'' may be added to return the specified number of nearby cities (the ''Duplicates'' option must also be used for the duplicate tags to be returned). May include regular expressions for more flexible matching of city names. See <https://exiftool.org/geolocation.html> for more details. Default is undef.

GeolocAltNames

Flag to search alternate Geolocation city names if available (ie. if `$Image::ExifTool::Geolocation::altDir` has been set). Set to 0 to disable use of the alternate names. Default is 1.

GeolocFeature

Comma-separated list of feature codes to include in city search, or exclude if the list begins with a dash (−). Valid feature codes are PPL, PPLA, PPLA2, PPLA3, PPLA4, PPLA5, PPLC, PPLCH, PPLF, PPLG, PPLH, PPLL, PPLQ, PPLR, PPLS, PPLW, PPLX, STLMT and Other, plus possible user-include codes if an alternate database is used. See <http://www.geonames.org/export/codes.html#P> for a description of these codes. Default is undef.

GeolocMaxDist

Maximum distance in km to the geolocation city. The Geolocation tags are not generated if the distance is greater than this. Default is undef.

GeolocMinPop

Minimum population for the Geolocation city. Cities smaller than this are ignored. Default is undef.

GeoMaxIntSecs

Maximum interpolation time in seconds for geotagging. Geotagging is treated as an extrapolation if the Geotime value lies between two fixes in the same track which are separated by a number of seconds greater than this. Otherwise, the coordinates are calculated as a linear interpolation between the nearest fixes on either side of the Geotime value. Set to 0 to disable interpolation and use the coordinates of the nearest fix instead (provided it is within GeoMaxExtSecs, otherwise geotagging fails). Default is 1800.

GeoMaxExtSecs
: Maximum extrapolation time in seconds for geotagging. Geotagging fails if the Geotime value lies outside a GPS track by a number of seconds greater than this. Otherwise, for an extrapolation the coordinates of the nearest fix are taken (ie. it is assumed that you weren't moving during this period). Default is 1800.

GeoMaxHDOP
: Maximum Horizontal (2D) Dilution Of Precision for geotagging. GPS fixes are ignored if the HDOP is greater than this. Default is undef.

GeoMaxPDOP
: Maximum Position (3D) Dilution Of Precision for geotagging. GPS fixes are ignored if the PDOP is greater than this. Default is undef.

GeoHPosErr
: Expression to set GPSHPositioningError based on value of GPSDOP when geotagging. For example, use `'$GPSDOP * 4'` to set GPSHPositioningError to 4 times GPSDOP.

GeoMinSats
: Minimum number of satellites for geotagging. GPS fixes are ignored if the number of acquired satellites is less than this. Default is undef.

GeoSpeedRef
: Reference units for writing GPSSpeed when geotagging:

```
'K', 'k' or 'km/h'  - km/h
'M', 'm' or 'mph'   - mph
<anything else>     - knots (default undef)
```

GeoUserTag
: Additional user-defined tags to write when geotagging from GPX or CSV files. Format is '*TAG=TOKEN*,...' where *TAG* is an ExifTool tag name (with optional group name), and *TOKEN* is is the GPX property name or CSV column name. eg) `Location=desc` will write the Location tag based on the value of 'desc' in the GPS log file. The tag value is taken from the nearest fix in time. Multiple tags may be specified using comma separators. Default is undef.

GlobalTimeShift
: Time shift to apply to all extracted date/time PrintConv values. Does not affect ValueConv values. Value is a date/time shift string (see **Image::ExifTool::Shift**(3pm)) with a leading '-' for negative shifts, or a tag name with option group prefix followed by '+' or '-' then the shift string. Default is undef.

  Note: When specifying a number of months and/or years to shift, the tag for the starting date should be specified so the number of days can be determined unambiguously. For example:

```
'createdate-1:0:0 0:0:0' - shift back by the length of the
                           year before the CreateDate value
'xmp:createdate+0:2:0 0' - shift forward by the length of
                           the 2 months after XMP:CreateDate
```

  If the starting tag is not specified, or the specified tag isn't available, then the shift is calculated based on the first shifted tag.

GPSQuadrant
: This option is used to specify the GPS quadrant in the case where a warning was issued because the GPS quadrant couldn't be determined. The value is a 2-character code where the first character is 'N' or 'S' and the second character is 'E' or 'W' (case insensitive). If this option is not set and the quadrant is unknown, a warning is issued and the quadrant is assumed to be 'NE'.

Group#
: Extract tags only for specified groups in family # (Group0 assumed if # not given). The option value may be a single group name or a reference to a list of groups. Case is significant in group

names. Specify a group to be excluded by preceding group name with a '−'. See ''GetGroup'' for a description of group families, and ''GetAllGroups [static]'' for lists of group names.

HexTagIDs
: Return hexadecimal instead of decimal for the family 7 group names of tags with numerical ID's.

HtmlDump
: Dump information in hex to dynamic HTML web page. The value may be 0−3 for increasingly larger limits on the maximum block size. Default is 0. Output goes to the file specified by the TextOut option (*STDOUT by default).

HtmlDumpBase
: Base for HTML dump offsets. If not defined, the EXIF/TIFF base offset is used. Set to 0 for absolute offsets. Default is undef.

IgnoreGroups
: Comma-separated list of group names to ignore when reading. The group names are case insensitive and may be preceded by a family number. Set to undef to clear the previous IgnoreGroups list. Default is undef.

IgnoreMinorErrors
: Flag to ignore minor errors. Causes minor errors to be downgraded to warnings, and minor warnings to be ignored. This option is provided mainly to allow writing of files when minor errors occur, but by ignoring some minor warnings the behaviour of ExifTool may be changed to allow some questionable operations to proceed (such as extracting thumbnail and preview images even if they don't have a recognizable header). Minor errors and warnings are denoted by ''[minor]'' at the start of the message, or ''[Minor]'' (with a capital ''M'') for warnings that affect processing when ignored.

IgnoreTags
: Comma-separated list of tag names to ignore when reading. This may help in situations where memory is limited because the ignored tag values are not stored in memory. The tag names are case insensitive and group names and wildcards are not allowed. A special tag name of ''All'' may be used to ignore all tags except those specified by the ''RequestTags'' option. Set to undef to clear the previous IgnoreTags list. Default is undef.

ImageHashType
: Sets type of hash algorithm used for the ImageDataHash tag calculation. Supported options are 'MD5', 'SHA256', and 'SHA512'. Default is 'MD5'.

KeepUTCTime
: Flag to keep UTC times in Zulu time zone instead of converting to local time. Affects only times which are stored as seconds since the UTC epoch. Default is undef.

Lang
: Localized language for exiftool tag descriptions, etc. Available languages are given by the Image::ExifTool::Lang module names (eg. 'fr', 'zh_cn'). If the specified language isn't available, the option is not changed. May be set to undef to select the built-in default language. Default is 'en'.

LargeFileSupport
: Flag to indicate that 64−bit file offsets are supported on this system. If not set, processing is aborted if a chunk larger than 2 GB is encountered. Set to 1 to process large chunks, or 2 to process with a warning. Default is 1.

LimitLongValues
: When extracting values for some specific tags (usually Unknown tags), the PrintConv values are length-limited and the value is truncated with an ellipsis (''[...]'') if it exceeds a specified length. This option specifies the length limit for these tags. A setting of 4 or less disables the limit (because the ellipsis string is longer than this). Default is 60.

ListItem
: Return only a specific item from list-type values. A value of 0 returns the first item in the list, 1 return the second item, etc. Negative indices may also be used, with −1 representing the last item in the list. Applies only to the top-level list of nested lists. Default is undef to return all items in the list.

ListJoin
: Separator used to join the PrintConv value of multi-item List-type tags into a single string. If not defined, multi-item lists are returned as a list reference. Does not affect ValueConv values. Default is ', '.

ListSplit
: Regular expression used to split values of list-type tags into individual items when writing. (eg. use ',\\s*' to split a comma-separated list.) Split when writing either PrintConv or ValueConv values. Default is undef.

MakerNotes
: Option to extract MakerNotes and other writable subdirectories (such as PrintIM) as a data block. Normally when the MakerNotes are extracted they are rebuilt to include data outside the boundaries of the original maker note data block, but a value of 2 disables this feature. Possible values are:

```
0 - Do not extract writable subdirectories (same as default of undef)
1 - Extract and rebuild maker notes into self-contained block
2 - Extract without rebuilding maker notes
```

MDItemTags
: Flag to extract the OS X metadata item tags (see the ''mdls'' man page and ''MacOS MDItem Tags'' in Image::ExifTool::TagNames for more information).

MissingTagValue
: Value for missing tags in tag name expressions (or tags where the advanced formatting expression returns undef). If not set, a minor error is issued for missing values, or the value is set to '' if ''IgnoreMinorErrors'' is set. Default is undef.

NoDups
: Flag to remove duplicate items from queued values for List-type tags when writing. This applies only to queued values, and doesn't resolve duplicates with existing values in the file when adding to an existing list. Default is undef.

NoMandatory
: Flag to bypass writing of mandatory EXIF tags. Default is undef.

NoMultiExif
: Raise error when attempting to write multi-segment EXIF in a JPEG image. Default is undef.

NoPDFList
: Flag to avoid splitting PDF list-type tag values into separate items. Default is undef.

NoWarning[+]
: Regular expression to suppress matching warning messages. For example, a value of ''^Ignored'' suppresses all warnings that begin with the word ''Ignored''. Has no other effect on processing, unlike IgnoreMinorWarnings for some warnings. Start the expression with ''(?i)'' for case-insensitive matching. Use NoWarning+ to add to existing expressions. Default is undef.

Password
: Password for reading/writing password-protected PDF documents. Ignored if a password is not required. Character encoding of the password is determined by the value of the Charset option at processing time. Default is undef.

Plot

Settings to override defaults for SVG plot feature. Value is a comma delimited string of settings, or undef to revert to default settings. Settings are accumulated if this option is set multiple times. Commas in the settings must be escaped as '',''. Valid settings and their default values are:

```
"Type=Line"           - plot type (Line, Scatter or Histogram)
"Style=Line"          - data style (Line, Marker and/or Fill)
"NBins=20"            - number of bins for histogram plot
"Size=800 600"        - width,height of output image
"Margin=60 15 15 30"  - left,top,right,bottom margins around plot area
"Legend=0 0"          - x,y offset to shift plot legend
"TxtPad=10 10"        - padding between text and x,y scale
"LineSpacing=20"      - spacing between text lines
"Stroke=1"            - plot stroke width and marker-size scaling
Title, XLabel, YLabel - plot title and x/y axis labels (no default)
XMin, XMax            - x axis minimum/maximum (autoscaling if not set)
YMin, YMax            - y axis minimum/maximum
Multi                 - flag to draw multiple plots, one for each dataset
Split                 - flag to split strings of numbers into lists
                          (> 1 to split into lists of N items)
"Grid=darkgray"       - grid color
"Text=black"          - color of text and plot border
"Bkg="                - background color (default is transparent)
"Cols=red green blue black orange gray fuchsia brown turquoise gold"
                      - colors for plot data
"Marks=circle square triangle diamond star plus pentagon left down right"
                      - marker-shape names for each dataset
```

PrintConv

Flag to enable automatic print conversion. Also enables inverse print conversion for writing. Default is 1.

PrintCSV

Flag to directly print CSV-format output rather than extracting tags normally. Currently this feature applies only to GM PDR data (see ''GM Tags'' in Image::ExifTool::TagNames). Setting this option automatically sets ''IgnoreTags'' to ''all'', and ''ExtractEmbedded'' to 1. Default is undef.

QuickTimeHandler

Flag set to add an 'mdir' Handler to a newly created Meta box when adding QuickTime ItemList tags. Adobe Bridge does not add this Handler, but it is commonly found in samples from other software, and it has been reported that Apple QuickTime Player and Photos.apps will ignore ItemList tags if this is missing. Default is 1.

QuickTimePad

Flag to preserve the padding of some QuickTime atoms when writing. QuickTime-based Canon CR3 files pad the values of container atoms with null bytes. This padding is removed by default when the file is rewritten, but setting this option to 1 adds padding to preserve the original atom size if the new atom would be smaller than the original. Default is undef.

QuickTimeUndLang

Flag to use 0x55c5 ('und') instead of 0x0000 for an undefined language code. This is a work-around because it has been reported that Apple Photos may sometimes not recognize the zero language code.

QuickTimeUTC

Flag set to assume that QuickTime date/time values are stored as UTC, causing conversion to local time when they are extracted and from local time when written. According to the QuickTime specification date/time values should be UTC, but many digital cameras store local

time instead (presumably because they don't know the time zone), so the default is to not convert these times (except for Canon CR3 files, which always use UTC times). This option also disables the auto-detection of incorrect time-zero offsets in QuickTime date/time values, and enforces a time zero of 1904 as per the QuickTime specification.

RequestAll
: Flag to request all tags to be extracted. This causes some tags to be generated which normally would not be unless specifically requested (by passing the tag name to ''ImageInfo'' or ''ExtractInfo''). May be set to 2 or 3 to enable generation of some additional tags as mentioned in the tag name documentation. Default is undef.

RequestTags
: List of additional tag and/or group names to request in subsequent calls to ''ExtractInfo''. This option is useful only for tags/groups which aren't extracted unless specifically requested, or if ''IgnoreTags'' is set to ''All''. Value may be a list reference, a delimited string of names (any delimiter is allowed), or undef to clear the current RequestTags list. Groups are requested by adding a colon after the name (eg. ''MacOS:''). Names are converted to lower case as they are added to the list. Default is undef.

SaveBin
: Flag to save binary values of tags, accessible through calls to ''GetValue'' with a value type of ''Bin''.

SaveFormat
: Flag to save EXIF/TIFF format type as the family 6 group name when extracting information. Without this option set, the family 6 group names are not generated. Default is undef. See ''GetGroup'' for more details.

SavePath
: Flag to save the metadata path as the family 5 group name when extracting information. Without this option set, the family 5 group names are not generated. Default is undef. See the ''GetGroup'' option for more details.

ScanForXMP
: Flag to scan all files (even unrecognized formats) for XMP information unless XMP was already found in the file. When combined with the FastScan option, only unrecognized file types are scanned for XMP. Default is undef.

Sort
: Specifies order to sort tags in returned list:

```
Input  - Sort in same order as input tag arguments (default)
File   - Sort in order that tags were found in the file
Tag    - Sort alphabetically by tag name
Descr  - Sort by tag description (for current Lang setting)
Group# - Sort by tag group, where # is zero or more family
         numbers separated by colons. If # is not specified,
         Group0 is assumed.  See GetGroup for a description
         of group families.
```

Sort2
: Secondary sort order used for tags within each group when Sort is 'Group':

```
File   - Sort in order tags were found in the file (default)
Tag    - Sort alphabetically by tag name
Descr  - Sort by tag description (for current Lang setting)
```

StrictDate
: Flag to return undefined value for any date which can't be converted when the DateFormat option is used. Default is undef.

```
undef − Same as 0 for reading/writing, or 1 for copying
  0   − Return date/time value unchanged if it can't be converted
  1   − Return undef if date/time value can't be converted
```

When set to 1 while writing a PrintConv date/time value with the DateFormat option set, the value is written only if POSIX::strptime or Time::Piece is available and can successfully convert the value.

For PNG CreationTime, a setting of 1 has the additional effect of causing the date/time to be reformatted according to PNG 1.2 recommendation (RFC−1123) when writing, and a warning to be issued for any non-standard value when reading (but note that Windows may not recognize PNG date/time values in standard format).

Struct

Flag to return XMP structures as hash references instead of flattening into individual tags. Has no effect when writing since both flattened and structured tags may always be written. A special ‘‘_ordered_keys_’’ element containing a list of ordered keys may exist if the structure elements are ordered (see the OrderedKeys method). Possible values are:

```
undef − (default) Same as 0 for reading, 2 for copying
  0   − Read/copy flattened tags
  1   − Read/copy structured tags
  2   − Read/copy both flattened and structured tags, but flag
        flattened tags as 'unsafe' for copying
```

StructFormat

Format for serialized structures when reading/writing.

```
undef − Default ExifTool format
JSON  − JSON format
JSONQ − JSON with quoted numerical values
```

Note that the JSONQ setting causes all values in the exiftool application −json output to be quoted, regardless of whether or not they are in a structure.

SystemTags

Flag to extract the following additional File System tags: FileAttributes, FileDeviceNumber, FileInodeNumber, FileHardLinks, FileUserID, FileGroupID, FileDeviceID, FileBlockSize and FileBlockCount.

SystemTimeRes

Number of digits in sub-seconds for system times and some other epoch times. A value of 0 rounds to the nearest second. Negative values cause trailing zeros to be removed. eg) A value of −6 returns system times to 6 decimal places with trailing zeros removed. Default is 0, which rounds to the nearest second. Note that round-off errors may appear starting at the 6th decimal place for some time formats.

TextOut

Output file reference for Verbose and HtmlDump options. Default is *STDOUT.

TimeZone

Set the time zone for local date/time values. The value is a time zone offset like ‘‘+05:00’’ (but note that the offset is to UTC, not from UTC, so it is positive for western time zones), or a time zone name like ‘‘EST5EDT’’. For Unix-based systems, the value may also be a time zone ID like ‘‘America/New_York’’. Requires Time::Piece on Windows, or POSIX::tzset on other systems. Default is undef.

UndefTags

Flag to maintain undefined tag values in the application **−if** expression when the **−f** or **−m** option is also used. Default is undef.

Unknown

Flag to get the values of unknown tags. If set to 1, unknown tags are extracted from EXIF (or other tagged-format) directories. If set to 2, unknown tags are also extracted from binary data blocks. Default is 0.

UserParam

Special option to set/get user-defined parameters. Useful to allow external input into tag name expressions and ValueConv logic. Valid UserParam values are:

```
PARAM          - Get parameter
PARAM=         - Clear parameter
PARAM^=        - Set parameter to empty string
PARAM=VALUE    - Set parameter
<hash ref>     - Set entire UserParam hash lookup
undef          - Clear all user parameters
```

Where *PARAM* is the user-defined parameter name (case insensitive).

User-defined parameters may be accessed in tag name expressions by prefixing the parameter name with a dollar sign just like normal tags, or via the API by calling `Options('UserParam','PARAM')`. Appending a hash tag (#) to the parameter name also causes the parameter to be extracted as a normal tag (in the UserParam group). If called without additional arguments, `Options('UserParam')` returns a reference to the hash of all user parameters (with lower-case names).

Validate

Flag to perform extra validation metadata checks when reading, causing extra warnings to be generated if problems are found. Default is undef.

Verbose

Print verbose messages to file specified by TextOut option. Value may be from 0 to 5 for increasingly verbose messages. Default is 0. With the verbose option set, messages are printed to the console as the file is parsed. Level 1 prints the tag names and raw values. Level 2 adds more details about the tags. Level 3 adds a hex dump of the tag data, but with limits on the number of bytes dumped. Levels 4 and 5 remove the dump limit on tag values and JPEG segment data respectively.

WindowsLongPath

Support long path names in Windows. Enabling this option automatically enables the WindowsWideFile feature. Default is 1 if Win32::API is available, or 0 otherwise.

WindowsWideFile

Force the use of wide-character Windows I/O functions. This may be necessary when files are on a network drive and the current directory name contains Unicode characters. Without this option the wide-character functions are used only if the specified file path contains Unicode characters.

WriteMode

Set tag write/create mode. Value is a string of one or more characters from list below. Default is 'wcg'.

```
w - Write existing tags
c - Create new tags
g - create new Groups as necessary
```

The level of the group differs for different types of metadata. For XMP or IPTC this is the full XMP/IPTC block (the family 0 group), but for EXIF this is the individual IFD (the family 1 group). The 'w' and 'c' modes are tested only when ''SetNewValue'' is called, but the 'g' mode is also tested in ''WriteInfo''.

XAttrTags
: Flag to extract the OS X extended attribute tags (see the ‘‘xattr’’ man page and ‘‘MacOS XAttr Tags’’ in Image::ExifTool::TagNames for more information).

XMPAutoConv
: Flag to enable automatic conversion for unknown XMP tags with values that look like rational numbers or dates. Default is 1.

Return Values:
: The original value of the last specified parameter.

**ClearOptions**

Reset all options to their default values. Loads user-defined default option values from the `%Image::ExifTool::UserDefined::Options` hash in the .ExifTool_config file if it exists.

```
$exifTool->ClearOptions();
```

Inputs:
: 0) ExifTool object reference

Return Values:
: (none)

**ExtractInfo**

Extract all meta information from an image. Need not be called when ‘‘ImageInfo’’ is used.

```
$success = $exifTool->ExtractInfo('image.jpg', \%options);
```

Inputs:
: ‘‘ExtractInfo’’ takes exactly the same arguments as ‘‘ImageInfo’’. The only difference is that a list of tag keys is not returned if an ARRAY reference is given.

Return Value:
: 1 if this was a recognized file format, 0 otherwise (and ’Error’ tag set).

**GetInfo**

‘‘GetInfo’’ is called to return meta information after it has been extracted from the image by a previous call to ‘‘ExtractInfo’’ or ‘‘ImageInfo’’. This function may be called repeatedly after a single call to ‘‘ExtractInfo’’ or ‘‘ImageInfo’’.

```
# get image width and height only
$info = $exifTool->GetInfo('ImageWidth', 'ImageHeight');

# get all Error and Warning messages
$info = $exifTool->GetInfo('Error', 'Warning');

# get information for all tags in list (list updated with tags found)
$info = $exifTool->GetInfo(\@ioTagList);

# get all information in Author or Location groups
$info = $exifTool->GetInfo({Group2 => ['Author', 'Location']});
```

Inputs:
: Inputs are the same as ‘‘ExtractInfo’’ and ‘‘ImageInfo’’ except that an image can not be specified.

Return Value:
: Reference to information hash, the same as with ‘‘ImageInfo’’.

**WriteInfo**

Write meta information to a file. The specified source file is rewritten to the same-type destination file with new information as specified by previous calls to ‘‘SetNewValue’’. The necessary segments and/or directories are created in the destination file as required to store the specified information. May be called repeatedly to write the same information to additional files without the need to call ‘‘SetNewValue’’ again.

ExifTool queues all new values that are assigned via calls to ‘‘SetNewValue’’, then applies them to any number of files through one or more calls to ‘‘WriteInfo’’. These queued values may be accessed through ‘‘GetNewValue’’, and are completely separate from metadata extracted from files via ‘‘ExtractInfo’’ or ‘‘ImageInfo’’ and accessed through ‘‘GetInfo’’ or ‘‘GetValue’’.

To be clear, it is NOT necessary to call ‘‘ExtractInfo’’ or ‘‘ImageInfo’’ before ‘‘WriteInfo’’. ‘‘WriteInfo’’ changes only metadata specified by previous calls to ‘‘SetNewValue’’.

```
# add information to a source file, writing output to new file
$exifTool->WriteInfo($srcfile, $dstfile);

# create XMP data file from scratch
$exifTool->WriteInfo(undef, $dstfile, 'XMP');

# overwrite file (you do have backups, right?)
$exifTool->WriteInfo($srcfile);
```

Inputs:

0) ExifTool object reference

1) Source file name, file reference, scalar reference, or undef to create a file from scratch. A reference to a File::RandomAccess object is also allowed as a source, but in this case the destination is not optional.

2) [optional] Destination file name, file reference, scalar reference to write to memory, or undef to overwrite the original file. May be ’-’ to write to stdout.

3) [optional] Destination file type. Ignored if a source is defined.

Return Value:

1 if file was written OK, 2 if file was written but no changes made, 0 on file write error.

If an error code is returned, an Error tag is set and GetValue(’Error’) can be called to obtain the error description. A Warning tag may be set even if this routine is successful. Calling WriteInfo clears any pre-existing Error and Warning tags.

```
$errorMessage = $exifTool->GetValue('Error');
$warningMessage = $exifTool->GetValue('Warning');
```

Notes:

The source file name may be undefined to create a file from scratch (currently only XMP, MIE, ICC, VRD, DR4, EXV and EXIF files can be created in this way — see ‘‘CanCreate’’ for details). If undefined, the destination file type is required unless the type can be determined from the extension of the destination file name.

If a destination file name is given, the specified file must not exist because an existing destination file will not be overwritten. Any new values for FileName, Directory or HardLink are ignored when a destination file name is specified.

The destination file name may be undefined to overwrite the original file (make sure you have backups!). In this case, if a source file name is provided, a temporary file is created and renamed to replace the source file if no errors occurred while writing. Otherwise, if a source file reference or scalar reference is used, the image is first written to memory then copied back to replace the original if there were no errors.

On Mac OS systems, the file resource fork is preserved if this routine is called with a source file name.

**GetTagList**

Get a sorted list of tags from the specified information hash or tag list.

```
@tags = $exifTool->GetTagList($info, 'Group0');
```

Inputs:

0) ExifTool object reference

1) [optional] Information hash reference or tag list reference

2) [optional] Sort order (’Input’, ’File’, ’Tag’, ’Descr’ or ’Group#’)

3) [optional] Secondary sort order (’File’, ’Tag’ or ’Descr’)

If the information hash or tag list reference is not provided, then the list of found tags from the last call to ‘‘ImageInfo’’, ‘‘ExtractInfo’’ or ‘‘GetInfo’’ is used instead, and the result is the same as if ‘‘GetFoundTags’’ was called. If sort order is not specified, the sort order is taken from the current options settings.

Return Values:

A list of tag keys in the specified order.

**GetFoundTags**

Get list of found tags in specified sort order. The found tags are the tags for the information obtained from the most recent call to ‘‘ImageInfo’’, ‘‘ExtractInfo’’ or ‘‘GetInfo’’ for this object.

```
@tags = $exifTool->GetFoundTags('File');
```

Inputs:

0) ExifTool object reference

1) [optional] Sort order (’Input’, ’File’, ’Tag’, ’Descr’ or ’Group#’)

2) [optional] Secondary sort order (’File’, ’Tag’ or ’Descr’)

If sort order is not specified, the sort order from the ExifTool options is used.

Return Values:

A list of tag keys in the specified order.

**GetRequestedTags**

Get list of requested tags. These are the tags that were specified in the arguments of the most recent call to ‘‘ImageInfo’’, ‘‘ExtractInfo’’ or ‘‘GetInfo’’, including tags specified via a tag list reference. Shortcut tags are expanded in the list.

```
@tags = $exifTool->GetRequestedTags();
```

Inputs:

(none)

Return Values:

List of requested tag keys in the same order that the tags were specified. Note that this list will be empty if tags were not specifically requested (ie. If extracting all tags).

**GetValue**

Get the value of a specified tag. The returned value is either the human-readable (PrintConv) value, the converted machine-readable (ValueConv) value, the original raw (Raw) value, or the original rational (Rational) value for rational formats. If the value type is not specified, the PrintConv value is returned if the PrintConv option is set, otherwise the ValueConv value is returned. The PrintConv values are same as the values returned by ‘‘ImageInfo’’ and ‘‘GetInfo’’ in the tag/value hash unless the PrintConv option is disabled.

Tags which represent lists of multiple values (as may happen with ’Keywords’ for example) are handled specially. In scalar context, the returned PrintConv value for these tags is either a string of values or a list reference (depending on the ListJoin option setting), and the ValueConv value is always a list reference. But in list context, ‘‘GetValue’’ always returns the list itself.

```
# PrintConv example
my $val = $exifTool->GetValue($tag);
if (ref $val eq 'SCALAR') {
    print "$tag = (unprintable value)\n";
} else {
    print "$tag = $val\n";
}

# ValueConv examples
my $val = $exifTool->GetValue($tag, 'ValueConv');
if (ref $val eq 'ARRAY') {
    print "$tag is a list of values\n";
} elsif (ref $val eq 'SCALAR') {
    print "$tag represents binary data\n";
} else {
    print "$tag is a simple scalar\n";
}

my @keywords = $exifTool->GetValue('Keywords', 'ValueConv');
```

Inputs:

0) ExifTool object reference

1) Tag key, or case-insensitive tag name with optional group prefix(es)

2) [optional] Value type: 'PrintConv', 'ValueConv', 'Both', 'Raw', 'Bin' or 'Rational'

The default value type is 'PrintConv' if the PrintConv option is set, otherwise the default is 'ValueConv'. A value type of 'Both' returns both ValueConv and PrintConv values as a list. 'Raw' returns the raw decoded tag value. 'Bin' returns the original binary data for EXIF tags if the ''SaveBin'' option was set. 'Rational' returns the raw rational value as a string fraction for rational types, or undef for other types.

Return Values:

The value of the specified tag. If the tag represents a list of multiple values and the ListJoin option is enabled then PrintConv returns a string of values, otherwise a reference to the list is returned in scalar context. The list itself is returned in list context. (Unless 'Both' values are requested, in which case two list references are returned, regardless of context.) Values may also be scalar references to binary data, or hash references if the ''Struct'' option is set.

Note: It is possible for ''GetValue'' to return an undefined ValueConv or PrintConv value (or an empty list in list context) even if the tag exists, since it is possible for these conversions to yield undefined values. And the Rational value will be undefined for any non-rational tag. The Raw value should always exist if the tag exists.

**SetNewValue**

Set the new value for a tag. The routine may be called multiple times to set the values of many tags before using ''WriteInfo'' to write the new values to an image. These values remain queued for writing to subsequent files until ''SetNewValue'' is called without arguments to reset the queued values.

For list-type tags (like Keywords), either call repeatedly with the same tag name for each value, or call with a reference to the list of values.

```
# set a new value for a tag (errors go to STDERR)
$success = $exifTool->SetNewValue($tag, $value);

# set a new value and capture any error message
($success, $errStr) = $exifTool->SetNewValue($tag, $value);
```

```
# delete information for specified tag if it exists in image
# (also resets AddValue and DelValue options for this tag)
$exifTool->SetNewValue($tag);

# reset all values from previous calls to SetNewValue()
$exifTool->SetNewValue();

# delete a specific keyword
$exifTool->SetNewValue('Keywords', $word, DelValue => 1);

# set keywords (a list-type tag) with two new values
$exifTool->SetNewValue(Keywords => 'word1');
$exifTool->SetNewValue(Keywords => 'word2');
# equivalent, but set both in one call using an array reference
$exifTool->SetNewValue(Keywords => ['word1','word2']);

# add a keyword without replacing existing keywords in the file
$exifTool->SetNewValue(Keywords => $word, AddValue => 1);

# conditionally add a tag if it didn't exist before,
# or replace it if it had a specified value ("old value")
$exifTool->SetNewValue(Description => '', DelValue => 1);
$exifTool->SetNewValue(Description => 'old value', DelValue => 1);
$exifTool->SetNewValue(Description => 'new value');

# set a tag in a specific group
$exifTool->SetNewValue(Headline => $val, Group => 'XMP');
$exifTool->SetNewValue('XMP:Headline' => $val);  # (equivalent)

# shift original date/time back by 2.5 hours
$exifTool->SetNewValue(DateTimeOriginal => '2:30', Shift => -1);

# write a tag only if it had a specific value
# (the order of the following calls is not significant)
$exifTool->SetNewValue(Title => $oldVal, DelValue => 1);
$exifTool->SetNewValue(Title => $newVal);

# write tag by numerical value
$exifTool->SetNewValue(Orientation => 6, Type => 'ValueConv');
$exifTool->SetNewValue('Orientation#' => 6);  # (equivalent)

# delete all but EXIF tags
$exifTool->SetNewValue('*');  # delete all...
$exifTool->SetNewValue('EXIF:*', undef, Replace => 2); # ...but EXIF

# write structured information as a HASH reference
$exifTool->SetNewValue('XMP:Flash' => {
    mode   => 'on',
    fired  => 'true',
    return => 'not'
});

# write structured information as a serialized string
$exifTool->SetNewValue('XMP:Flash'=>'{mode=on,fired=true,return=not}');
```

(See <https://exiftool.org/struct.html#Serialize> for a description of the structure serialization technique.)

Inputs:

0) ExifTool object reference

1) [optional] Tag key or tag name, or undef to clear all new values. The tag name may be prefixed by one or more family 0, 1 or 2 group names with optional leading family numbers, separated by colons (eg. 'EXIF:Artist', 'XMP:Time:*'), which is equivalent to using a Group option argument. Also, a '#' may be appended to the tag name (eg. 'EXIF:Orientation#'), with the same effect as setting Type to 'ValueConv'. Wildcards ('*' and '?') may be used in the tag name to assign or delete multiple tags simultaneously. A tag name of '*' is special when deleting information, and will delete an entire group even if some individual tags in the group are not writable, but only if a single family 0 or 1 group is specified (otherwise the tags are deleted individually). Use ''GetDeleteGroups'' to get a list of deletable group names, and see Image::ExifTool::TagNames for a complete list of tag names.

2) [optional] New value for tag. Undefined to delete tag from file. May be a scalar, scalar reference, list reference to set a list of values, or hash reference for a structure. Integer values may be specified as a hexadecimal string (with a leading '0x'), and simple rational values may be specified in fractional form (eg. '4/10'). Structure tags may be specified either as a hash reference or a serialized string (see the last two examples above).

3−N) [optional] SetNewValue option/value pairs (see below).

SetNewValue Options:

AddValue
: Specifies that the value be added to an existing list in a file rather than overwriting the existing values. Valid settings are 0 (overwrite any existing tag value), 1 (add to an existing list and warn for non-list tags) or 2 (add to existing list and overwrite non-list tags). Default is 0.

DelValue
: Delete existing tag from a file if it has the specified value. For list-type tags this deletes a specified item from the list. For non-list tags this may be used to conditionally replace a tag by providing a new value in a separate call to SetNewValue (see examples above). For structured tags, the entire structure is deleted/replaced only if all of the specified fields match the existing structure. Option values are 0 or 1. Default is 0.

EditGroup
: Create tags in existing groups only. Don't create new group. Valid values are 0 and 1. Effectively removes the 'g' from the ExifTool WriteMode option for this tag only. Default is 0.

EditOnly
: Edit tag only if it already exists. Don't create new tag. Valid values are 0 and 1. Effectively removes the 'c' from the ExifTool WriteMode option for this tag only. Default is 0.

Group
: Specifies group name where tag should be written. This option is superseded by any group specified in the tag name. If not specified, tag is written to highest priority group as specified by ''SetNewGroups''. May be one or more family 0, 1 or 2 groups with optional leading family number, separated by colons. Case is not significant.

NoFlat
: Treat flattened tags as 'unsafe'.

NoShortcut
: Disables default behaviour of looking up tag in shortcuts if not found otherwise.

Protected
: Bit mask for tag protection levels to write. Bit 0x01 allows writing of 'unsafe' tags (ie. tags not copied automatically via ''SetNewValuesFromFile''). Bit 0x02 allows writing of 'protected' tags, and should only be used internally by ExifTool. See Image::ExifTool::TagNames, for a list of tag names indicating 'unsafe' and 'protected' tags. Default is 0.

ProtectSaved
: Avoid setting new values which were saved after the Nth call to ''SaveNewValues''. Has no effect on unsaved values, or values saved before Nth call. Option value is N. Default is undef.

Replace
: Flag to replace the previous new values for this tag (ie. replace the values set in previous calls to ''SetNewValue''). This option is most commonly used to replace previously-set new values for list-type tags. Valid values are 0 (set new value normally — adds to new values for list-type tags), 1 (reset any previous new values before setting new value) or 2 (reset previous new values only; new value argument is ignored). Default is 0.

Shift
: Shift the tag by the specified value. Currently only date/time tags and tags with numerical values may be shifted. Undefined for no shift, 1 for a positive shift, or −1 for a negative shift. A value of 0 causes a positive shift to be applied if the tag is shiftable and AddValue is set, or a negative shift for date/time tags only if DelValue is set. Default is undef. See **Image::ExifTool::Shift** (3pm) for more information.

Type
: The type of value being set. Valid values are PrintConv, ValueConv or Raw. Default is PrintConv if the ''PrintConv'' Option is set, otherwise ValueConv.

Return Values:
: In scalar context, returns the number of tags set and error messages are printed to STDERR. In list context, returns the number of tags set, and the error string (which is undefined if there was no error).

Notes:
: When deleting groups of tags, the Replace option may be used to exclude specific groups from a mass delete. However, this technique may not be used to exclude individual tags from a group delete (unless a family 2 group was specified in the delete). Instead, use ''SetNewValuesFromFile'' to recover the values of individual tags after deleting a group.

: When deleting all tags from a JPEG image, the APP14 ''Adobe'' information is not deleted by default because doing so may affect the appearance of the image. However, this information may be deleted by specifying it explicitly, either by group (with 'Adobe:*') or as a block (with 'Adobe').

**GetNewValue**

Get the new Raw value for a tag. This is the value set by ''SetNewValue'' this is queued to be written to file. List-type tags may return multiple values in list context.

```
$rawVal = $exifTool->GetNewValue($tag);

@rawVals = $exifTool->GetNewValue($tag);
```

Notes:
: The API NoDups option applies when this routine is called, and removes duplicate items from values returned for List-type tags.

Inputs:
: 0) ExifTool object reference

: 1) Tag name (case sensitive, may be prefixed by family 0, 1 or 7 group names, separated by colons)

Return Values:
: List of new Raw tag values, or first value in list when called in scalar context. The list may be empty either if the tag isn't being written, or if it is being deleted (ie. if ''SetNewValue'' was called without a value).

**SetNewValuesFromFile**

A very powerful routine that sets new values for tags from information found in a specified file.

```
# set new values from all information in a file...
my $info = $exifTool->SetNewValuesFromFile($srcFile);
# ...then write these values to another image
my $result = $exifTool->WriteInfo($file2, $outFile);

# set all new values, preserving original groups
$exifTool->SetNewValuesFromFile($srcFile, '*:*');

# set specific information
$exifTool->SetNewValuesFromFile($srcFile, @tags);

# set new value from a different tag in specific group
$exifTool->SetNewValuesFromFile($fp, 'XMP-dc:Subject<IPTC:Keywords');

# add all IPTC keywords to XMP subject list
$exifTool->SetNewValuesFromFile($fp, 'XMP-dc:Subject+<IPTC:Keywords');

# set new value from an expression involving other tags
$exifTool->SetNewValuesFromFile($file,
    'Comment<ISO=$ISO Aperture=$aperture Exposure=$shutterSpeed');

# set keywords list from the values of multiple tags
$exifTool->SetNewValuesFromFile($file, { Replace => 0 },
    'keywords<xmp:subject', 'keywords<filename');

# copy all EXIF information, preserving the original IFD
# (without '*.*<' tags would be copied to the preferred EXIF IFD)
$exifTool->SetNewValuesFromFile($file, '*:*<EXIF:*');

# copy all tags with names starting with "gps" (note: this is
# different than "gps:*" because it will also copy XMP GPS tags)
$exifTool->SetNewValuesFromFile($file, 'gps*');

# set FileName from Model, translating questionable characters
$exifTool->SetNewValuesFromFile($file,
    'filename<${model; tr(/\\\\?*:|"><)(_) }.jpg');
```

Inputs:

0) ExifTool object reference

1) File name, file reference, or scalar reference

2-N) [optional] List of tag names to set or options hash references. All writable tags are set if none are specified. The tag names are not case sensitive, and may be prefixed by one or more family 0, 1, 2 or 7 group names with optional leading family numbers, separated by colons (eg. 'exif:iso'). A leading '-' indicates tags to be excluded (eg. '-comment'), or a trailing '#' causes the ValueConv value to be copied (same as setting the Type option to 'ValueConv' for this tag only). A leading '+' sets the Replace option to 0 on a per-tag basis (see Options below). Wildcards ('*' and '?') may be used in the tag name. A tag name of '*' is commonly used when a group is specified to copy all tags in the group (eg. 'XMP:*').

A special feature allows tag names of the form 'DSTTAG<SRCTAG' (or 'SRCTAG>DSTTAG') to be specified to copy information to a tag with a different name or a specified group. Both 'SRCTAG' and 'DSTTAG' may contain wildcards and/or be prefixed by a group name (eg. 'fileModifyDate<modifyDate' or 'xmp:*<*'), and/or suffixed by a '#' to disable print conversion. Copied tags may also be added or deleted from a list with arguments of the form 'DSTTAG+<SRCTAG' or 'DSTTAG-<SRCTAG'. Tags are evaluated in order, so exclusions apply

only to tags included earlier in the list. An extension of this feature allows the tag value to be set from a string containing tag names with leading '$' symbols (eg. 'Comment<the file is `$filename`'). Braces '{}' may be used around a tag name to separate it from subsequent text, and a '$$' is used to to represent a '$' symbol. The behaviour for missing tags in expressions is defined by the ''MissingTagValue'' option. The tag value may be modified via changes to the default input variable ($_) in a Perl expression placed inside the braces and after a semicolon following the tag name (see the last example above). A '@' may be added after the tag name (before the semicolon) to make the expression act on individual list items instead of the concatenated string for list-type tags. The expression has access to the full ExifTool API through the current ExifTool object (`$self`) and the tag key (`$tag`). Braces within the expression must be balanced.

Multiple options hash references may be passed to set different options for different tags. Options apply to subsequent tags in the argument list.

By default, this routine will commute information between same-named tags in different groups, allowing information to be translated between images with different formats. This behaviour may be modified by specifying a group name for extracted tags (even if '*' is used as a group name), in which case the information is written to the original group, unless redirected to a different group. When '*' is used for a group name, by default the family 1 group of the original tag is preserved, but a different family may be specified with a leading family number. (For example, specifying '*:*' copies all information while preserving the original family 1 groups, while '0*:*' preserves the family 0 group.)

SetNewValuesFromFile Options:

The options are the same was for ''SetNewValue'', and are passed directly to ''SetNewValue'' internally, with a few exceptions:

− The Replace option defaults to 1 instead of 0 as with ''SetNewValue'', however the tag name argument may be prefixed with '+' to set the Replace option to 0 for this argument only.

− The AddValue or DelValue option is set for individual tags if '+>' or '−>' (or '+<' or '−<') are used.

− The Group option is set for tags where a group name is given.

− The Protected flag is set to 1 for individually specified tags.

− The Type option also applies to extracted tags.

Return Values:

A hash of information that was set successfully. May include Warning or Error entries if there were problems reading the input file.

Notes:

The PrintConv option applies to this routine, but it normally should be left on to provide more reliable transfer of information between groups.

If a preview image exists, it is not copied. The preview image must be transferred separately if desired, in a separate call to ''WriteInfo''

When simply copying all information between files of the same type, it is usually desirable to preserve the original groups by specifying '*:*' for the tags to set.

The ''Duplicates'' option is always in effect for tags extracted from the source file using this routine.

The ''Struct'' option is enabled by default for tags extracted by this routine. This allows the hierarchy of complex structures to be preserved when copying, but the Struct option may be set to 0 to override this behaviour and copy as flattened tags instead.

**CountNewValues**

Return the total number of new values set.

```
$numSet = $exifTool->CountNewValues();
($numSet, $numPseudo) = $exifTool->CountNewValues();
```

Inputs:

0) ExifTool object reference

Return Values:

In scalar context, returns the total number of tags with new values set. In list context, also returns the number of ‘‘pseudo’’ tag values which have been set. ‘‘Pseudo’’ tags are tags like FileName and FileModifyDate which are not contained within the file and can be changed without rewriting the file.

**SaveNewValues**

Save state of new values to be later restored by ‘‘RestoreNewValues’’.

```
$exifTool->SaveNewValues();         # save state of new values
$exifTool->SetNewValue(ISO => 100); # set new value for ISO
$exifTool->WriteInfo($src, $dst1);  # write ISO + previous new values
$exifTool->RestoreNewValues();      # restore previous new values
$exifTool->WriteInfo($src, $dst2);  # write previous new values only
```

Inputs:

0) ExifTool object reference

Return Value:

Count of the number of times this routine has been called (N) since the last time the new values were reset.

**RestoreNewValues**

Restore new values to the settings that existed when ‘‘SaveNewValues’’ was last called. May be called repeatedly after a single call to ‘‘SaveNewValues’’. See ‘‘SaveNewValues’’ above for an example.

Inputs:

0) ExifTool object reference

Return Value:

None.

**SetAlternateFile**

Specify alternate file from which to read metadata. Tags from the alternate file are available after ‘‘ExtractInfo’’ is called or during a call to ‘‘SetNewValuesFromFile’’ by using a family 8 group name (eg. ’File1’ in the example below).

```
$exifTool->SetAlternateFile(File1 => 'images/test1.jpg');
```

Inputs:

0) ExifTool object reference

1) Family 8 group name, case insensitive (eg. ’File1’, ’File2’...)

2) Name of alternate input file, or undef to reset

Return Values:

1 on success, or 0 if the group name is invalid.

**SetFileModifyDate**

Write the filesystem modification or creation time from the new value of the FileModifyDate or FileCreateDate tag.

```
$exifTool->SetNewValue(FileModifyDate => '2000:01:02 03:04:05-05:00',
                       Protected => 1);
$result = $exifTool->SetFileModifyDate($file);
```

Inputs:

0) ExifTool object reference

1) File name

2) [optional] Base time if applying shift (days before $^T)

3) [optional] Tag to write: 'FileModifyDate' (default), or 'FileCreateDate'

Return Value:

1 if the time was changed, 0 if nothing was done, or −1 if there was an error setting the time.

Notes:

Equivalent to, but more efficient than calling ''WriteInfo'' when only the FileModifyDate or FileCreateDate tag has been set. If a timezone is not specified, local time is assumed. When shifting, the time of the original file is used unless the optional base time is specified.

The ability to write FileCreateDate is currently restricted to Windows systems only.

**SetFileName**

Set the file name and directory, or create a hard link. If not specified, the new file name is derived from the new values of the FileName and Directory tags, or from the HardLink or SymLink tag if creating a link. If the FileName tag contains a '/', then the file is renamed into a new directory. If FileName ends with '/', then it is taken as a directory name and the file is moved into the new directory. The new value for the Directory tag takes precedence over any directory specified in FileName.

```
$result = $exifTool->SetFileName($file);
$result = $exifTool->SetFileName($file, $newName);
```

Inputs:

0) ExifTool object reference

1) Current file name

2) [optional] New file name

3) [optional] 'HardLink' or 'SymLink' to create a hard or symbolic link instead of renaming the file, or 'Test' to test renaming feature by printing the old and new names instead of changing anything.

Return Value:

1 on success, 0 if nothing was done, or −1 if there was an error renaming the file or creating the link.

Notes:

Will not overwrite existing files. New directories are created as necessary. If the file is successfully renamed, the new file name may be accessed via `$$exifTool{NewName}`.

**SetNewGroups**

Set the order of the preferred groups when adding new information. In subsequent calls to ''SetNewValue'', new information will be created in the first valid group of this list. This has an impact only if the group is not specified when calling ''SetNewValue'' and if the tag name exists in more than one group. The default order is EXIF, IPTC, XMP, MakerNotes, QuickTime, Photoshop, ICC_Profile, CanonVRD, Adobe. Any family 0 group name may be used. Case is not significant.

```
$exifTool->SetNewGroups('XMP','EXIF','IPTC');
```

Inputs:

0) ExifTool object reference

1−N) Groups in order of priority. If no groups are specified, the priorities are reset to the defaults.

Return Value:

None.

**GetNewGroups**

Get current group priority list.

```
@groups = $exifTool->GetNewGroups();
```

Inputs:

0) ExifTool object reference

Return Values:

List of group names in order of write priority. Highest priority first.

**GetTagID**

Get the ID for the specified tag. The ID is the IFD tag number in EXIF information, the property name in XMP information, or the data offset in a binary data block. For some tags, such as Composite tags where there is no ID, an empty string is returned. In list context, also returns a language code for the tag if available and different from the default language (eg. with alternate language entries for XMP ''lang-alt'' tags).

```
$id = $exifTool->GetTagID($tag);
($id, $lang) = $exifTool->GetTagID($tag);
```

Inputs:

0) ExifTool object reference

1) Tag key

Return Values:

In scalar context, returns the tag ID or '' if there is no ID for this tag. In list context, returns the tag ID (or '') and the language code (or undef).

**GetDescription**

Get description for specified tag. This function will always return a defined value. In the case where the description doesn't exist, one is generated from the tag name.

Inputs:

0) ExifTool object reference

1) Tag key

Return Values:

A description for the specified tag.

**GetGroup**

Get group name(s) for a specified tag.

```
# return family 0 group name (eg. 'EXIF');
$group = $exifTool->GetGroup($tag, 0);

# return all groups (eg. qw{EXIF IFD0 Author Main})
@groups = $exifTool->GetGroup($tag);

# return groups as a string (eg. 'Main:IFD0:Author')
$group = $exifTool->GetGroup($tag, ':3:1:2');

# return groups as a simplified string (eg. 'IFD0:Author')
$group = $exifTool->GetGroup($tag, '3:1:2');
```

Inputs:

0) ExifTool object reference

1) Tag key

2) [optional] Group family number, or string of numbers separated by colons

Return Values:

Group name (or '' if tag has no group). If no group family is specified, ''GetGroup'' returns the name of the group in family 0 when called in scalar context, or the names of groups for all families in list context. Returns a string of group names separated by colons if the input group family contains a colon. The string is simplified to remove a leading 'Main:' and adjacent identical group names unless the family string begins with a colon.

Notes:

The group family numbers are currently available:

```
0) Information Type         (eg. EXIF, XMP, IPTC)
1) Specific Location        (eg. IFD0, ExifIFD, XMP-dc)
2) Category                 (eg. Author, Time)
3) Document Number          (eg. Main, Doc1, Doc3-2)
4) Instance Number          (eg. Copy1, Copy2, Copy3)
5) Metadata Path            (eg. JPEG-APP1-IFD0-ExifIFD)
6) EXIF/TIFF Format         (eg. int8u, int32u, undef, string)
7) Tag ID                   (eg. ID-271, ID-rights, ID-a9aut)
8) Alternate File Number    (eg. File1, File2, File3)
```

Families 0 and 1 are based on the file structure, and are similar except that family 1 is more specific and sub-divides some groups to give more detail about the specific location where the information was found. For example, the EXIF group is split up based on the specific IFD (Image File Directory), the MakerNotes group is divided into groups for each manufacturer, and the XMP group is separated based on the XMP namespace prefix. Note that only common XMP namespaces are listed in the GetAllGroups documentation, but additional namespaces may be present in some XMP data. Also note that the 'XMP-xmp...' group names may appear in the older form 'XMP-xap...' since these names evolved as the XMP standard was developed. The ICC_Profile group is broken down to give information about the specific ICC_Profile tag from which multiple values were extracted. As well, information extracted from the ICC_Profile header is separated into the ICC-header group.

Family 2 classifies information based on the logical category to which the information refers.

Family 3 gives the document number for tags extracted from embedded documents, or 'Main' for tags from the main document. (See the ''ExtractEmbedded'' option for extracting tags from embedded documents.) Nested sub-documents (if they exist) are indicated by numbers separated with dashes in the group name, to an arbitrary depth. (eg. 'Doc2-3-1' is the 1st sub-sub-document of the 3rd sub-document of the 2nd embedded document of the main file.) Document numbers are also used to differentiate samples for timed metadata in videos.

Family 4 provides a method for differentiating tags when multiple tags exist with the same name in the same location. The primary instance of a tag (the tag extracted when the Duplicates option is disabled and no group is specified) has no family 4 group name, but additional instances have family 4 group names of 'Copy1', 'Copy2', 'Copy3', etc. For convenience, the primary tag may also be accessed using a group name of 'Copy0'.

Family 5 is experimental, and gives the complete path for the metadata in the file. Generated only if the ''SavePath'' option is used when extracting.

Family 6 is currently used only for EXIF/TIFF metadata, and gives the format type of the extracted value. Generated only if the ''SaveFormat'' option is used when extracting.

Family 7 is used for tag ID's. The group names are the actual tag ID's, with a leading ''ID-'' string. Non-numerical ID's have characters other than [-_A-Za-z0-9] converted to hex. Numerical tag ID's are returned in hex if the ''HexTagIDs'' option is set, otherwise decimal is used. When specifying a family 7 group name, numerical ID's may be in hex or decimal, and non-numerical ID's may or may not have characters other than [-_A-Za-z0-9] converted to hex. Note that unlike other group names, the tag ID's of family 7 group names are case sensitive (but the leading ''ID-'' is not).

Family 8 specifies the alternate file set from a call to ''SetAlternateFile''.

See ''GetAllGroups [static]'' for complete lists of group names.

**GetGroups**

Get list of group names that exist in the specified information.

```
@groups = $exifTool->GetGroups($info, 2);
@groups = $exifTool->GetGroups('3:1');
```

Inputs:

0) ExifTool object reference

1) [optional] Info hash ref (default is all extracted info)

2) [optional] Group family number, or string of numbers (default 0)

Return Values:

List of group names in alphabetical order. If information hash is not specified, the group names are returned for all extracted information. See ''GetGroup'' for an description of family numbers and family number strings.

**BuildCompositeTags**

Builds composite tags from required tags. The composite tags are convenience tags which are derived from the values of other tags. This routine is called automatically by ''ImageInfo'' and ''ExtractInfo'' if the Composite option is set.

Inputs:

0) ExifTool object reference

Return Values:

(none)

Notes:

Tag values are calculated in alphabetical order unless a tag Require's or Desire's another composite tag, in which case the calculation is deferred until after the other tag is calculated.

Composite tags may need to read data from the image for their value to be determined, and for these ''BuildCompositeTags'' must be called while the image is available. This is only a problem if ''ImageInfo'' is called with a filename (as opposed to a file reference or scalar reference) since in this case the file is closed before ''ImageInfo'' returns. Here the Composite option may be used so that ''BuildCompositeTags'' is called from within ''ImageInfo'', before the file is closed.

**AvailableOptions [static]**

Get a list of available API options. (See ''Options'' for option details.)

Inputs:

(none)

Return Values:

Reference to list of available options. Each entry in the list is a list reference with 4 items: 0=Option name, 1=Default value, 2=Description, 3=flag set if option is undocumented.

```
my $opts = Image::ExifTool::<b>AvailableOptions</b>();
foreach (@$opts) {
    my ($optionName, $defaultValue, $description) = @$_;
    ...
}
```

**GetTagName [static]**

Get name of tag from tag key. This is a convenience function that strips the embedded instance number, if it exists, from the tag key.

Note: ''static'' in the heading above indicates that the function does not require an ExifTool object reference as the first argument. All functions documented below are also static.

```
$tagName = Image::ExifTool::GetTagName($tag);
```

Inputs:

0) Tag key

Return Value:

Tag name. This is the same as the tag key but has the instance number removed.

**GetShortcuts [static]**

Get a list of shortcut tags.

Inputs:

(none)

Return Values:

List of shortcut tags (as defined in Image::ExifTool::Shortcuts).

**GetAllTags [static]**

Get list of all available tag names.

```
@tagList = Image::ExifTool::GetAllTags($group);
```

Inputs:

0) [optional] Group name, or string of group names separated by colons

Return Values:

A list of all available tags in alphabetical order, or all tags in a specified group or intersection of groups. The group name is case insensitive, and any group in families 0−2 may be used except for EXIF family 1 groups (ie. the specific IFD).

**GetWritableTags [static]**

Get list of all writable tag names.

```
@tagList = Image::ExifTool::GetWritableTags($group);
```

Inputs:

0) [optional] Group name, or string of group names separated by colons

Return Values:

A list of all writable tags in alphabetical order. These are the tags for which values may be set through ''SetNewValue''. If a group name is given, returns only writable tags in specified group(s). The group name is case insensitive, and any group in families 0−2 may be used except for EXIF family 1 groups (ie. the specific IFD).

**GetAllGroups [static]**

Get list of all group names in specified family.

```
@groupList = Image::ExifTool::GetAllGroups($family);
```

Inputs:

0) Group family number (0−7)

Return Values:

A list of all groups in the specified family in alphabetical order.

Here is a complete list of groups for each of these families:

Family 0 (Information Type):

AAC, AFCP, AIFF, APE, APP0, APP1, APP10, APP11, APP12, APP13, APP14, APP15, APP2, APP3, APP4, APP5, APP6, APP7, APP8, APP9, ASF, Audible, Canon, CanonVRD, Composite, DICOM, DjVu, DNG, Ducky, DV, EXE, EXIF, ExifTool, File, FITS, FLAC, Flash, FlashPix, FLIR, Font, FotoStation, Garmin, GeoTiff, GIF, GIMP, GM, GoPro, H264, HTML, ICC_Profile, ID3, IPTC, ISO, ITC, JFIF, JPEG, Jpeg2000, JSON, JUMBF, Kandao, Leaf, LNK, Lytro, M2TS, MakerNotes, Matroska, Meta, MIE, MIFF, MISB, MNG, MOI, MPC, MPEG, MPF, MXF, Ogg, OpenEXR, Opus, Palm, PanasonicRaw, Parrot, PDF, PhotoCD, PhotoMechanic, Photoshop, PICT, PLIST, PNG, PostScript, PrintIM, Protobuf, PSP, QuickTime, Radiance, RAF, Rawzor, Real, Red, RIFF, RSRC, RTF, SigmaRaw, Sony, Stim, SVG, Theora, Torrent, Trailer, VCard, Vorbis, WTV, XML, XMP, ZIP

Family 1 (Specific Location):

AAC, AADAccelFeatures, AC3, AccelData, Activity, Adobe, AdobeCM, AdobeDNG, AFCP, AIFF, AntChannelID, AntRx, AntTx, APE, APP2, Apple, AROT, ASF, Audible, AudioItemList, AudioKeys, AudioUserData, AuxBattery, AVI1, AviationAttitude, Barometer, BeatIntervals, BikeProfile, BloodPress, CadenceZone, CameraEvent, CameraIFD, Canon, CanonCustom, CanonDR4, CanonRaw, CanonVRD, Capabilities, Casio, CBOR, Chapter#, ChronoShotData, ChronoShotSession, CIFF, ClimbPro, Composite, Connectivity, Course, CoursePoint, DevDataID, DeviceInfo, DeviceSettings, DICOM, DiveAlarm, DiveApneaAlarm, DiveGas, DiveSettings, DiveSummary, DJI, DjVu, DjVu-Meta, DNG, Ducky, DV, EPPIM, Event, ExdDataConceptConfig, ExdDataFieldConfig, ExdScreenConfig, EXE, ExerciseTitle, EXIF, ExifIFD, ExifTool, FieldCaps, FieldDescr, File, FileCaps, FileCreator, FileID, FITS, FLAC, Flash, FlashPix, FLIR, Font, FotoStation, FujiFilm, FujiIFD, Garmin, GE, GeoTiff, GIF, GIMP, GlobParamIFD, GM, Goal, Google, GoPro, GPS, GraphConv, GSpherical, GyroData, H264, HP, HR, HRMProfile, HRV, HRVStatusSummary, HRVValue, HRZone, HSA_HRData, HSA_SPO2Data, HSAAccelData, HSABodyBattery, HSAConfigData, HSAEvent, HSAGyroData, HSARespirData, HSAStepData, HSAStressData, HSAWristTemp, HTC, HTML, HTML-dc, HTML-ncc, HTML-office, HTML-prod, HTML−vw96, HTTP-equiv, ICC-chrm, ICC-cicp, ICC-clrt, ICC-header, ICC-meas, ICC-meta, ICC-view, ICC_Profile, ICC_Profile#, ID3, ID3v1, ID3v1_Enh, ID3v2_2, ID3v2_3, ID3v2_4, IFD0, IFD1, InfiRay, Insta360, InteropIFD, IPTC, IPTC#, ISO, ITC, ItemList, iTunes, JFIF, JFXX, JPEG, JPEG-HDR, Jpeg2000, JPS, JSON, JUMBF, Jump, JVC, KDC_IFD, Keys, KFIX, Kodak, KodakBordersIFD, KodakEffectsIFD, KodakIFD, KVAR, KyoceraRaw, Lap, Leaf, LeafSubIFD, Leica, Length, LNK, Lyrics3, Lytro, M−RAW, M2TS, MAC, MacOS, MagData, MakerNotes, MakerUnknown, Matroska, MaxMetData, MediaJukebox, MemoGlob, MesgCaps, Meta, MetaIFD, MetZone, Microsoft, MIE-Audio, MIE-Camera, MIE-Canon, MIE-Doc, MIE-Extender, MIE-Flash, MIE-Geo, MIE-GPS, MIE-Image, MIE-Lens, MIE-Main, MIE-MakerNotes, MIE-Meta, MIE-Orient, MIE-Preview, MIE-Thumbnail, MIE-Unknown, MIE-UTM, MIE-Video, MIFF, Minolta, MinoltaRaw, MISB, MNG, MOBI, MOI, MonitorHRData, MonitorInfo, Monitoring, Motorola, MPC, MPEG, MPF0, MPImage, MS-DOC, MXF, NapEvent, Nextbase, Nikon, NikonCapture, NikonCustom, NikonScan, NikonSettings, NineEdits, Nintendo, NITF, NMEA, OBDIIData, Ocad, Ogg, OHRSettings, Olympus, OneDSensorCal, OnePlus, OpenEXR, Opus, Palm, Panasonic, PanasonicRaw, Parrot, PDF, Pentax, PhaseOne, PhotoCD, PhotoMechanic, Photoshop, PICT, PictureInfo, PNG, PNG-cICP, PNG-pHYs, PostScript, PowerZone, PreviewIFD, PrintIM, ProfileIFD, PSP, Qualcomm, QuickTime, Radiance, RAF, RAF2, RawBBI, Rawzor, Real, Real-CONT, Real-MDPR, Real-PROP, Real−RA3, Real−RA4, Real−RA5, Real-RJMD, Reconyx, Record, Red, RespirationRate, Ricoh, RIFF, RMETA, RSRC, RTF, Samsung, Sanyo, Scalado, Schedule, SDMProfile, SEAL, SegFile, SegLap, SegLeaderboard, SegmentID, SegPoint, Session, Set, Sigma, SigmaRaw, SkinTempOvernight, SlaveDevice, SleepAssessment, SleepDisruptOvernight, SleepDisruptPeriod, SleepLevel, Software, Sony, SonyIDC, SpeedZone, SPIFF, Split, SplitSummary, SPO2Data, Sport, SR2, SR2DataIFD, SR2SubIFD, SRF#, Stim, StressLevel, SubIFD, SVG, System, TankSummary, TankUpdate, Theora, ThreeDSensorCal, TimeInZone, Torrent, Totals, Track#, Track#ItemList, Track#Keys, Track#UserData, TrainingFile, TrainingSettings, TSCorrelation, UserData, UserProfile, VCalendar, VCard, Version0, Video, VideoClip, VideoDescr, VideoFrame, VideoItemList, VideoKeys, VideoTitle, VideoUserData, Vivo, VNote, Vorbis, WatchfaceSettings, WeatherAlert, WeatherConditions, WeightScale, Workout, WorkoutSess, WorkoutStep, WTV, XML, XMP, XMP-aas, XMP-acdsee, XMP-acdsee-rs, XMP-album, XMP-apdi, XMP-apple-fi, XMP-ast, XMP-aux, XMP-cc, XMP-cell, XMP-crd, XMP-creatorAtom, XMP-crs, XMP-dc, XMP-Device, XMP-dex, XMP-DICOM, XMP-digiKam, XMP-drone-dji, XMP-dwc, XMP-et, XMP-exif, XMP-exifEX, XMP-expressionmedia, XMP-extensis, XMP-fpv, XMP-GAudio, XMP-GCamera, XMP-GContainer, XMP-GCreations, XMP-GDepth, XMP-getty, XMP-GFocus, XMP-GImage, XMP-GPano, XMP-GSpherical, XMP-hdr, XMP-HDRGainMap, XMP-hdrgm, XMP-ics, XMP-iptcCore, XMP-iptcExt, XMP-LImage, XMP-lr, XMP-mediapro, XMP-microsoft, XMP-MP, XMP−MP1, XMP-mwg-coll, XMP-mwg-kw, XMP-mwg-rs, XMP-nine, XMP-panorama, XMP-pdf, XMP-pdfx, XMP-photomech, XMP-photoshop, XMP-PixelLive, XMP-plus, XMP-pmi, XMP-prism, XMP-prl, XMP-prm, XMP-pur, XMP-rdf, XMP-sdc, XMP-seal, XMP-swf, XMP-tiff, XMP-x, XMP-xmp, XMP-xmpBJ, XMP-xmpDM, XMP-xmpDSA, XMP-xmpMM, XMP-xmpNote, XMP-xmpPLUS, XMP-xmpRights, XMP-xmpTPg, ZIP, ZonesTarget

Family 2 (Category):
Audio, Author, Camera, Device, Document, ExifTool, Image, Location, Other, Preview, Printing, Time, Unknown, Video

Family 3 (Document Number):
Doc#, Main

Family 4 (Instance Number):
Copy#

Family 5 (Metadata Path):
eg. JPEG−APP1−IFD0−ExifIFD

Family 6 (EXIF/TIFF Format):
int8u, string, int16u, int32u, rational64u, int8s, undef, int16s, int32s, rational64s, float, double, ifd, unicode, complex, int64u, int64s, ifd64

Family 7 (Tag ID):
ID-xxx (Where xxx is the tag ID. Numerical ID's are returned in hex with a leading ''0x'' if the HexTagIDs option is set, or decimal otherwise. Characters in non-numerical ID's which are not valid in a group name are returned as 2 hex digits.)

Family 8 (Alternate File):
File#

Note: This function may also be called as an ExifTool member function to allow the HexTagIDs option to be set when retrieving family 7 group names.

**GetDeleteGroups [static]**

Get list of all deletable group names.

```
@delGroups = Image::ExifTool::GetDeleteGroups();
```

Inputs:
None.

Return Values:
A list of deletable group names in alphabetical order. The current list of deletable group names is:

Adobe, AFCP, APP0, APP1, APP10, APP11, APP12, APP13, APP14, APP15, APP2, APP3, APP4, APP5, APP6, APP7, APP8, APP9, Audio, Author, Camera, CanonVRD, CIFF, Document, Ducky, EXIF, ExifIFD, ExifTool, File, FlashPix, FotoStation, GlobParamIFD, GPS, ICC_Profile, IFD0, IFD1, Image, Insta360, InteropIFD, IPTC, ItemList, JFIF, Jpeg2000, Keys, Location, MakerNotes, Meta, MetaIFD, Microsoft, MIE, MPF, NikonCapture, Other, PDF, PDF-update, PhotoMechanic, Photoshop, PNG, PNG-pHYs, Preview, PrintIM, Printing, QuickTime, RMETA, RSRC, SubIFD, Time, Trailer, UserData, Video, XML, XML−*, XMP, XMP−*

To schedule a group for deletion, call ''SetNewValue'' with a tag name like 'EXIF:*' and an undefined tag value.

Deleting a family 0 or 1 group will delete the entire corresponding block of metadata, but deleting a family 2 group (eg. Audio, Author, Camera, etc.) deletes the individual tags belonging to that category.

The 'Trailer' group allows all trailers in JPEG and TIFF-format images to be deleted at once, including unknown trailers. Note that the JPEG ''APP'' groups are special, and are used only to delete application segments which are not associated with another deletable group. For example, deleting 'APP14:*' will delete other APP14 segments, but not the APP14 ''Adobe'' segment.

**GetFileType [static]**

Get type of file given file name.

```
my $type = Image::ExifTool::GetFileType($filename);
my $desc = Image::ExifTool::GetFileType($filename, 1);
```

Inputs:

0) [optional] File name (or just an extension)

1) [optional] Flag to return a description instead of a type. Default is undef. Set to 0 to also return types of recognized but unsupported files (otherwise the return value for unsupported files is undef), or 1 to return descriptions.

Return Value:

A string, based on the file extension, which indicates the basic format of the file. Note that some files may be based on other formats (like many RAW image formats are based on TIFF). In list context, may return more than one file type if the file may be based on different formats. Returns undef if files with this extension are not yet supported by ExifTool. Returns a list of extensions for all supported file types if no input extension is specified (or all recognized file types if the description flag is set to 0). Returns a more detailed description of the specific file format when the description flag is set.

**CanWrite [static]**

Can the specified file be written?

```
my $writable = Image::ExifTool::CanWrite($filename);
```

Inputs:

0) File name or extension

Return Value:

True if ExifTool supports writing files of this type (based on the file extension).

**CanCreate [static]**

Can the specified file be created?

```
my $creatable = Image::ExifTool::CanCreate($filename);
```

Inputs:

0) File name or extension

Return Value:

True if ExifTool can create files with this extension from scratch. Currently, this can only be done with XMP, MIE, ICC, VRD, DR4, EXV and EXIF files.

**AddUserDefinedTags [static]**

Add user-defined tags to an existing tag table at run time. This differs from the usual technique of creating user-defined tags via the `%Image::ExifTool::UserDefined` hash (see the ExifTool_config file in the Image::ExifTool distribution) because it allows tags to be added after a tag table has been initialized.

```
use Image::ExifTool ':Public';
my %tags = (
    TestTagID1 => { Name => 'TestTagName1' },
    TestTagID2 => { Name => 'TestTagName2' },
);
my $num = AddUserDefinedTags('Image::ExifTool::PDF::Info', %tags);
```

Inputs:

0) Destination tag table name

1−N) Pairs of tag ID / tag information hash references for the new tags

Return Value:

The number of tags added.

Notes

Pre-existing tags with the same ID will be replaced in the destination table. See lib/Image/ExifTool/README in the full distribution for full details on the elements of the tag

information hash.

**OrderedKeys [static]**

Return a list of ordered keys from a tag value that is a HASH reference when the Struct option is used.

```
use Image::ExifTool ':Public';
my @keys = OrderedKeys($structRef);
```

Inputs:

0) Structure HASH reference

Return Value:

List of ordered keys, or sorted alphabetically if not ordered.

## CHARACTER ENCODINGS

Certain meta information formats allow coded character sets other than plain ASCII. When reading, most known encodings are converted to the external character set according to the ''Charset'' option, or to UTF-8 by default. When writing, the inverse conversions are performed. Alternatively, special characters may be converted to/from HTML character entities with the ''Escape'' HTML option.

A distinction is made between the external character set visible via the ExifTool API, and the internal character used to store text in the metadata of a file. These character sets may be specified separately as follows:

External Character Sets:

The encoding for tag values passed to/from ExifTool API functions is set via the ''Charset'' option, which is 'UTF8' by default.

The encoding of file names is specified via the ''CharsetFileName'' option. By default, ''CharsetFileName'' is not defined, and file names passed to ExifTool are used directly in calls to the system i/o routines (which expect UTF-8 strings on Mac/Linux, but default to the system code page on Windows). In this mode on Windows a warning is issued if a file name contains special characters, but this warning may be avoided by setting ''CharsetFileName'' to an empty string. Setting ''CharsetFileName'' to any other value causes file names to be converted from the specified encoding to one appropriate for the system. In Windows this also has the effect of activating Unicode filename support via the special Windows wide-character i/o routines if Win32API::File is available.

Note that setting the ''WindowsWideFile'' or ''WindowsLongPath'' option causes ''CharsetFileName'' to default to 'UTF8' in Windows if not defined, and ''WindowsLongPath'' is set by default if Win32::API is available.

Internal Character Sets:

The encodings used to store strings in the various metadata formats. These encodings may be changed for certain types of metadata via the ''CharsetEXIF'', ''CharsetID3'', ''CharsetIPTC'', ''CharsetPhotoshop'', ''CharsetQuickTime'' and ''CharsetRIFF'' options.

Values are returned as byte strings of encoded characters. Perl wide characters are not used. By default, most returned strings are encoded in UTF-8. For these, **Encode::decode_utf8()** may be used to convert to a sequence of logical Perl characters. Note that some settings of the PERL_UNICODE environment variable may be incompatible with ExifTool's character handling.

More specific details are given below about how character coding is handled for EXIF, IPTC, XMP, PNG, ID3, PDF, Photoshop, QuickTime, AIFF, MIE and Vorbis information:

**EXIF**

Most textual information in EXIF is stored in ASCII format (called ''string'' in the ExifTool tag name documentation). By default ExifTool does not convert these strings. However, it is not uncommon for applications to write UTF-8 or other encodings where ASCII is expected. To deal with these, ExifTool allows the internal EXIF string encoding to be specified with ''CharsetEXIF'', which causes EXIF string values to be converted from the specified character set when reading, and stored with this character set when writing. (The MWG recommends using UTF-8 encoding for EXIF strings, and in keeping with this the MWG module sets the default internal EXIF string encoding to UTF-8, but note that this will have no

effect unless the external encoding is also set to something other than the default of UTF−8.)

A few EXIF tags (UserComment, GPSProcessingMethod and GPSAreaInformation) support a designated internal text encoding, with values stored as ASCII, Unicode (UCS−2) or JIS. When reading these tags, ExifTool converts Unicode and JIS to the external character set specified by the ‘‘Charset’’ option, or to UTF−8 by default. ASCII text is not converted. When writing, text is stored as ASCII unless the string contains special characters, in which case it is converted from the external character set (UTF−8 by default), and stored as Unicode. ExifTool writes Unicode in native EXIF byte ordering by default, but the byte order may be specified by setting the ExifUnicodeByteOrder tag (see the Extra Tags documentation).

The EXIF ‘‘XP’’ tags (XPTitle, XPComment, etc) are always stored as little-endian Unicode (UCS−2), and are read and written using the specified character set.

**IPTC**

The value of the IPTC:CodedCharacterSet tag determines how the internal IPTC string values are interpreted. If CodedCharacterSet exists and has a value of ’UTF8’ (or ’ESC % G’) then string values are assumed to be stored as UTF−8, otherwise Windows Latin1 (cp1252, ’Latin’) coding is assumed by default, but this can be changed with the ‘‘CharsetIPTC’’ option. When reading, these strings are converted to the character set specified by the ‘‘Charset’’ option. When writing, the inverse conversions are performed. No conversion is done if the internal (IPTC) and external (ExifTool) character sets are the same. Note that ISO 2022 character set shifting is not supported. Instead, a warning is issued and the string is not converted if an ISO 2022 shift code is encountered. See <http://www.iptc.org/IIM/> for the official IPTC specification.

ExifTool may be used to convert IPTC values to a different internal encoding. To do this, all IPTC tags must be rewritten along with the desired value of CodedCharacterSet. For example, the following command changes the internal IPTC encoding to UTF−8 (from Windows Latin1 unless CodedCharacterSet was already ’UTF8’):

```
exiftool -tagsfromfile @ -iptc:all -codedcharacterset=utf8 a.jpg
```

or from Windows Latin2 (cp1250) to UTF−8:

```
exiftool -tagsfromfile @ -iptc:all -codedcharacterset=utf8 \
-charset iptc=latin2 a.jpg
```

and this command changes it back from UTF−8 to Windows Latin1 (cp1252):

```
exiftool -tagsfromfile @ -iptc:all -codedcharacterset= a.jpg
```

or to Windows Latin2:

```
exiftool -tagsfromfile @ -iptc:all -codedcharacterset= \
-charset iptc=latin2 a.jpg
```

Unless CodedCharacterSet is ’UTF8’, applications have no reliable way to determine the IPTC character encoding. For this reason, it is recommended that CodedCharacterSet be set to ’UTF8’ when creating new IPTC.

(Note: Here, ‘‘IPTC’’ Refers to the older IPTC IIM format. The more recent IPTC Core and Extension specifications actually use the XMP format.)

**XMP**

ExifTool reads XMP encoded as UTF−8, UTF−16 or UTF−32, and converts them all to UTF−8 internally. Also, all XML character entity references and numeric character references are converted. When writing, ExifTool always encodes XMP as UTF−8, converting the following 5 characters to XML character references: & < > ’ ". By default no further conversion is performed, however if the ‘‘Charset’’ option is other than ’UTF8’ then text is converted to/from the specified character set when reading/writing.

**PNG**

PNG TextualData tags are stored as tEXt, zTXt and iTXt chunks in PNG images. The tEXt and zTXt chunks use ISO 8859−1 encoding, while iTXt uses UTF−8. When reading, ExifTool converts all PNG textual data to the character set specified by the ‘‘Charset’’ option. When writing, ExifTool generates a tEXt chunk (or zTXt with the ‘‘Compress’’ option) if the text doesn’t contain special characters or if Latin encoding is

specified; otherwise an iTXt chunk is used and the text is converted from the specified character set and stored as UTF−8.

**JPEG Comment**

The encoding for the JPEG Comment (COM segment) is not specified, so ExifTool reads/writes this text without conversion.

**ID3**

The ID3v1 specification officially supports only ISO 8859−1 encoding (a subset of Windows Latin1), although some applications may incorrectly use other character sets. By default ExifTool converts ID3v1 text from Latin to the character set specified by the ‘‘Charset’’ option. However, the internal ID3v1 charset may be specified with the ‘‘CharsetID3’’ option. The encoding for ID3v2 information is stored in the file, so ExifTool converts ID3v2 text from this encoding to the character set specified by the ‘‘Charset’’ option. ExifTool does not currently write ID3 information.

**PDF**

PDF text strings are stored in either PDFDocEncoding (similar to Windows Latin1) or Unicode (UCS−2). When reading, ExifTool converts to the character set specified by the ‘‘Charset’’ option. When writing, ExifTool encodes input text from the specified character set as Unicode only if the string contains special characters, otherwise PDFDocEncoding is used.

**Photoshop**

Some Photoshop resource names are stored as Pascal strings with unknown encoding. By default, ExifTool assumes MacRoman encoding and converts this to UTF−8, but the internal and external character sets may be specified with the ‘‘CharsetPhotoshop’’ and ‘‘Charset’’ options respectively.

**QuickTime**

QuickTime text strings may be stored in a variety of poorly document formats. ExifTool does its best to decode these according to the ‘‘Charset’’ option setting. For some QuickTime strings, ExifTool assumes a default encoding of MacRoman, but this may be changed with the ‘‘CharsetQuickTime’’ option.

**AIFF**

AIFF strings are assumed to be stored in MacRoman, and are converted according to the ‘‘Charset’’ option when reading.

**RIFF**

The internal encoding of RIFF strings (eg. in AVI and WAV files) is assumed to be Latin unless otherwise specified by the RIFF CSET chunk or the ‘‘CharsetRIFF’’ option.

**MIE**

MIE strings are stored as either UTF−8 or ISO 8859−1. When reading, UTF−8 strings are converted according to the ‘‘Charset’’ option, and ISO 8859−1 strings are never converted. When writing, input strings are converted from the specified character set to UTF−8. The resulting strings are stored as UTF−8 if they contain multi-byte UTF−8 character sequences, otherwise they are stored as ISO 8859−1.

**Vorbis**

Vorbis comments are stored as UTF−8, and are converted to the character set specified by the ‘‘Charset’’ option.

## AUTHOR

Copyright 2003−2026, Phil Harvey

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself.

## ACKNOWLEDGEMENTS

Many people have helped in the development of ExifTool through their bug reports, comments and suggestions, and/or additions to the code. See the ACKNOWLEDGEMENTS in the individual Image::ExifTool modules and in html/index.html of the Image::ExifTool distribution package for a list of people who have contributed to this project.

## SEE ALSO

**exiftool** (1), **Image::ExifTool::TagNames** (3pm), **Image::ExifTool::Shortcuts** (3pm), **Image::ExifTool::Shift** (3pm), **Image::Info** (3pm), **Image::MetaData::JPEG** (3pm)