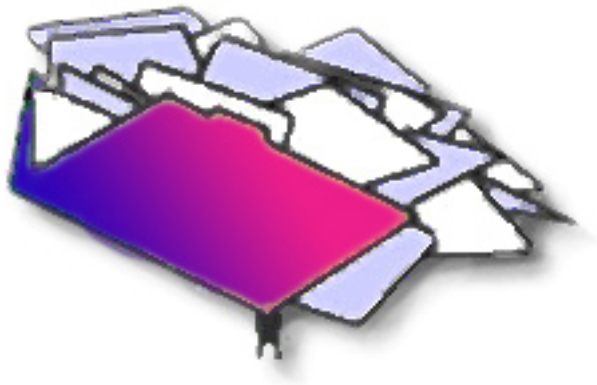


Troi File Plug-in 8.6
for FileMaker Pro 15
USER GUIDE

September 2016

Troi Automatisering
Boliviastraat 11
2408 MX Alphen a/d Rijn
The Netherlands

You can also visit the Troi web site at: http://www.troi.com for additional information.

Table of Contents

Table of Contents (continued)

Table of Contents (continued)

Installing plug-ins

Starting with FileMaker Pro 12 a plug-in can be installed directly from a container field. Please see the **EasyInstallTroiPlugins.fmp12** example file to install plug-ins with FileMaker Pro 12, 13 14 or 15.

The instructions below show FileMaker Pro 11 (no longer supported)

For Mac OS X:

- Quit FileMaker® Pro.
- Put the file “Troi_File.fmplugin” from the folder “Mac OS Plug-in” into the “Extensions” folder in the FileMaker Pro application folder.
- If you have installed previous versions of this plug-in, you are asked: “An older item named “Troi_File.fmplugin” already exists in this location. Do you want to replace it with the one you’re moving?’. Press the OK button.
- Start FileMaker Pro. The first time the Troi File Plug-in is used it will display a dialog box, indicating that it is loading and showing the registration status.

For Windows:

- Quit FileMaker Pro.
- Put the file "Troi_File_Plugin.fmx" from the directory "Windows" into the "Extensions" subdirectory in the FileMaker Pro application directory..
- If you have installed previous versions of this plug-in, you are asked: “This folder already contains a file called 'Troi_File_Plugin.fmx'. Would you like to replace the existing file with this one?’. Press the Yes button.
- Start FileMaker Pro. The Troi File Plug-in will display a dialog box, indicating that it is loading and showing the registration status.

TIP You can check which plug-ins you have loaded by going to the plug-in preferences: Choose **Preferences** from the **Edit** menu, and then choose **Plug-ins**.

You can now open the file "All File Examples.fmp12" to see how to use the plug-in's functions. There is also a function overview available.

If you have problems

This user guide tries to give you all the information necessary to use this plug-in. So if you have a problem please read this user guide first. Also you might visit our support web page:

http://www.troi.com/support/

This page contains FAQ's (Frequently Asked Questions), help on registration and much more. If that doesn't help you can get free support by email. Send your questions to **support@troi.com** with a full explanation of the problem. Also give as much relevant information (version of the plug-in, which platform, version of the operating system, version of FileMaker Pro) as possible. Note that due to spam we have to filter incoming email. It might happen that non-spam email is filtered out too. If you have sent an email and you don't get an answer, try to send another email, slightly differently formulated and include the word "FileMaker" in the body text.

If you find any mistakes in this manual or have a suggestion please let us know. We appreciate your feedback!

TIP You can get more information on returned error codes from our OSErrrs database on our web site: http://www.troi.com/software/oserrrs.html. This free FileMaker database lists all error codes for Windows and Mac OS X!

What can this plug-in do?

The Troi File Plug-in is a very powerful tool for getting access to information outside the FileMaker database. Any files stored on the rest of the computer can be accessed through the functions of the plug-in. All from within FileMaker you can:

- get data out of files on the disk of the computer
- create files anywhere on the hard disk and put data from FileMaker fields into it
- manipulate files and folders on the disk, like creating/deleting/copying/moving
- get metadata from images and movies, such as size, resolution, Exif IPTC etc.
- and much more...

Software requirements

System requirements for Mac OS X

Mac OS X 10.6.8 (Snow Leopard), Mac OS X 10.7 Lion, OS X 10.8 Mountain Lion, OS X 10.9 Mavericks, OS X 10.10 Yosemite, OS X 11.11 El Capitan, macOS Sierra (OS X 10.12).

System requirements for Windows

Windows 7 on Intel-compatible computer 1 GHz or faster
Windows 8, Windows 8.1
Windows 10

FileMaker Pro requirements

FileMaker Pro 12 or FileMaker Pro Advanced 12 or higher.
FileMaker Pro 13 or FileMaker Pro Advanced 13 or higher.
FileMaker Pro 14 or FileMaker Pro Advanced 14 or higher.
FileMaker Pro 15 or FileMaker Pro Advanced 15 or higher.

NOTE We have successfully tested it with FileMaker Pro 11 but we no longer provide active support for this version. Troi File plug-in will also probably run with FileMaker 7 to 10, but we have not tested this and we no longer provide support for this.

FileMaker Server requirements

FileMaker Server 12 or FileMaker Server Advanced 12 or higher.
FileMaker Server 13 or FileMaker Server Advanced 13 or higher.
FileMaker Server 14 or FileMaker Server Advanced 14 or higher.
FileMaker Server 15 or FileMaker Server Advanced 15 or higher.

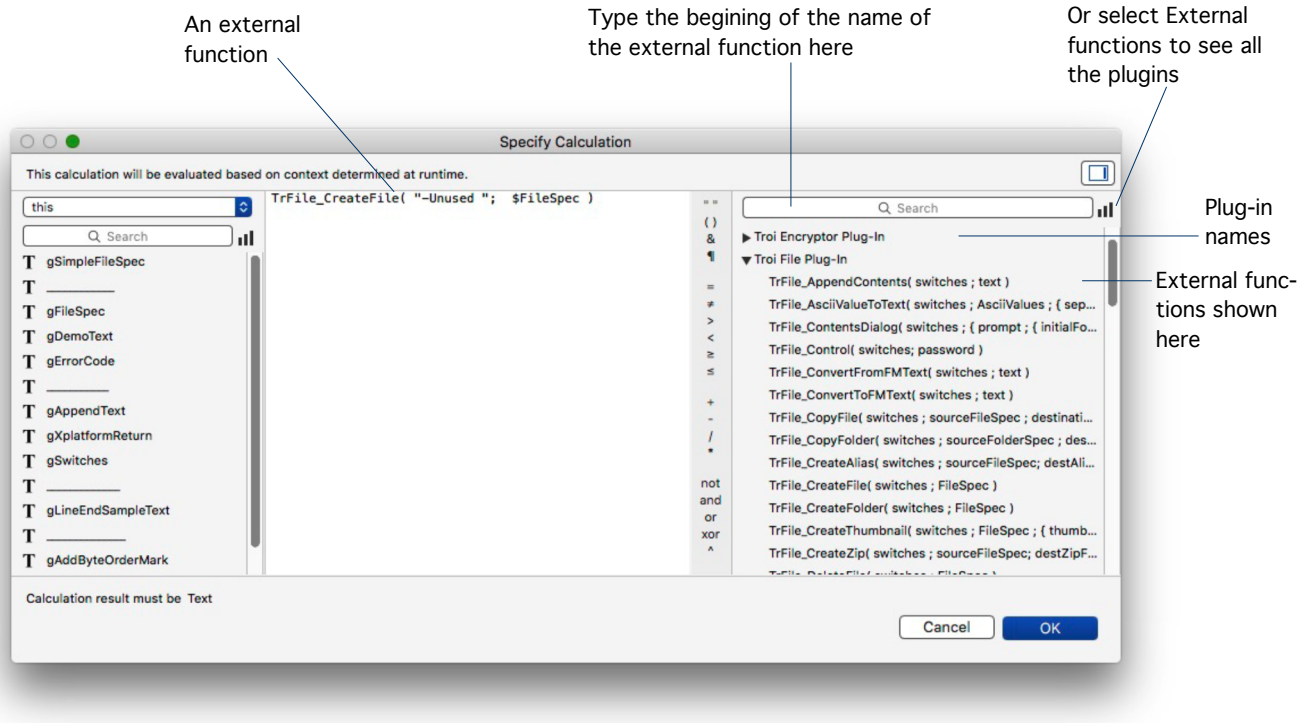
You can use FileMaker Server to serve databases that use functions of the Troi File Plug-in (client-side): You need to have the plug-in installed at the clients that use these functions.

Troi File Plug-in can also be used by FileMaker Server as a server-side plug-in or as a plug-in used by the web publishing engine. To use Troi Plug-ins as a server-side or web-side plug-in you need to purchase a special Server/Web license. More information can be found in the download or here:
http://www.troi.com/support/filemaker-server-side-plug-ins.html

Getting started

Using external functions

The Troi File Plug-in adds new functions to the standard functions that are available in FileMaker Pro. The functions added by a plug-in are called external functions. You can see those extra functions for all plug-ins at the top right of the Specify Calculation box:



You use special syntax with external functions: FunctionName(parameter1 ; parameter 2) where FunctionName is the name of an external function. A function can have zero or more parameters. Each parameter is separated by a semi-colon. Plug-ins don't work directly after installation. To access a plug-in's function, you need to add the calls to the function in a calculation, for example in a text calculation in Define Fields or in a script in Script Workspace (formaly called ScriptMaker).

Where to add the external functions?

Most of the external functions for this plug-in are intended to be used in a script step using a calculation, in a Set Field script step. For quite a lot of functions of this plug-in it makes no sense to add them to a define field calculation, as the functions will have side effects. When you are defining fields (choose Define Database from the File menu), only some of the functions will be visible. Those should be safe to use there, but be careful and think about before adding external functions to a calculation field.

About file paths and FileSpec's

To be able to specify a file or folder, the plug-in can use a full path or a FileMaker styled path.

NOTE Troi File Plug-in 8.0 no longer supports FSSpecs (see below).

On Windows the path is always a full path like this for a file:

C:\Data Files\Database\Test.Txt

and similar to this for a folder:

C:\My Files\Main\

If you use the FileMaker style paths this becomes:

filewin:/C:/Data Files/Database/Test.Txt

and similar to this for a folder:

filewin:/C:/My Files/Main/

On Mac OS X a FilePath can be a full path or a FileMaker styled path. A full path on Mac looks like this:

Mac HD:Data Files:Database:testfile

and similar to this for a folder:

Archive CD:My Files:Main:

If you use the FileMaker style paths on Mac OS X this becomes for a file:

filemac:/Mac HD/Data Files/Database/Test.Txt

and similar to this for a folder:

filemac:/Archive CD/My Files/Main/

Previous versions of Troi File Plug-in also supported a FSSpec, which has the following format:

:volumeID:directoryID:fileName.

An example would be:

:-1:300:testfile

NOTE 1 With Mac OS X FSSpec's are no longer supported. Therefore all plug-in functions now return a full path.
NOTE 2 On Mac OS X a volume name and therefore a full path need not be unique. Be careful with this.

Simple example

Say you want to get the contents of a report file that contains text. The file resides on a known place on the harddisk. Say you want the text to be put in the text field "myTextField". Create a script "Get Report Text". Add the following script step to this script:

Set Field[myTextField, TrFile_GetContents("-Unused" , "C:\DataFiles\Report.txt")]

If you run this script the contents of the file is put in the field "myTextField".

NOTE Function names, like TrFile_GetContents are not case sensitive.

As you can see the path "C:\DataFiles\Report.txt" in the script above makes it clear that this script only works on Windows. If you want to remove this platform dependency you can use a variable or global text field, which holds the path to the folder. Say the full path on Mac OS X would be "Disk:DataFiles:Report.txt". Also assume you have defined a global text field "gPathToFolder".

The script would then become like this:

```
If [Abs(Status(CurrentPlatform)) = 1]
        Comment[ ...on Mac OS X...]
        Set Field[gPathToFolder, "Disk:DataFiles:")]
Else
        Comment[ ...on Windows...]
        Set Field[gPathToFolder, "C:\DataFiles\")]
Endif
Set Field[myTextField, TrFile_GetContents( "-Unused" , gPathToFolder & "Report.txt")]
```

Of course the global gPathToFolder could be set earlier in a separate script, as part of the preferences.

Please take a close look at the included example files, as they provide a great starting point. From there you can move on, using the 60 functions of the plug-in as building blocks. Together they are a powerful tool set for working with files on the disk.

You can use globals or variables

With the release of FileMaker Pro 8 and later it is possible to use variables in calculations. Our example files in the download now both use global fields and variables to pass parameters and store the results of a plug-in function.

As this release of Troi File is intended for FileMaker Pro 12 and higher, we continue to move the scripts to use variables wherever possible. Note that not all examples are using variables yet.

All the plug-in functions work with variables just fine. For example if you have these script steps:

```
Set Field [this::gPath,  "your path here"  ]
Set Field [this::gImageDesc,  TrFile_MetaData(  "-GetImageDescription" ; this::gPath )  ]
```

With variables you can alternative use:

```
Set Variable [$Path,  "your path here"  ]
Set Variable [$ImageDesc,  TrFile_MetaData(  "-GetImageDescription" ; $Path )  ]
```

The main advantage of variables is that you don't need to define global fields that clutter your database definitions. The variables can stay local to the script.
See for example the new ManageMedia.fmp12 example file, which uses variables in its scripts. Note that this example file also depends on global fields, as the scripts have been copied from an older example file.

Summary of functions

The Troi File Plug-in adds the following functions to FileMaker Pro:

function name	short description
TrFile_AppendContents	append characters to the contents of an existing file
TrFile_AsciiValueToText	Converts numbers to their equivalent ASCII characters.
TrFile_ContentsDialog	shows the user a file selection dialog, contents of file is returned
TrFile_Control	controls (disable and enable)the functions of the plug-in
TrFile_ConvertFromFMText	converts text to text formatted in the format of the current platform
TrFile_ConvertToFMText	converts text to text formatted in the internal FileMaker format
TrFile_CreateAlias	creates an alias file (or shortcut) from the source file or source folder.
TrFile_CopyFile	copies a file to the specified path
TrFile_CopyFolder	copies a folder to the specified folder path.
TrFile_CreateFile	creates a new empty file
TrFile_CreateFolder	creates a new folder
TrFile_CreateZIP	creates ZIP file (compressed archive)
TrFile_CreateThumbnail	creates a thumbnail (a small image of normally 80x80 pixels) from an image
TrFile_DeleteFile	deletes a file
TrFile_DeleteFolder	deletes the folder indicated by the FolderSpec
TrFile_DiskInfo	retrieves information about a specified disk
TrFile_DragAndDrop	implements drag and drop of files to a FileMaker window
TrFile_ExecuteShell	execute a command in the (UNIX or Windows) shell of the operating system
TrFile_Exists	check for the existence of a file or folder
TrFile_FileSpecToFullPath	get full path of a file or folder from an FSSpec
TrFile_FindFolder	finds special folders
TrFile_FullPathToFileSpec	changes the full name path to the standard Mac OS FSSpec
TrFile_GetContents	get (a part of) the contents of the file
TrFile_GetDataSize	get the size of the data of a file
TrFile_GetDateCreated	get the creation date of a file
TrFile_GetDateModified	get the modification date of a file
TrFile_GetFileAttribute	returns an attribute of the file specified by the FileSpec
TrFile_GetFileCreator	get the Creator of a file
TrFile_GetFileHash	returns the MD5 hash value (checksum) of a file
TrFile_GetFileSize	get the file size of a file
TrFile_GetFileType	get the FileType of a file
TrFile_GetIcon	get the icon of a file and returns it as a PNG image.
TrFile_GetPathTo	returns the path to the FileMaker application or the current FileMaker file
TrFile_GetDateLastAccessed	returns the date a file was last accessed
TrFile_GetTimeLastAccessed	returns the time a file was last accessed
TrFile_GetTimestampLastAccessed	returns the timestamp when a file was last accessed
TrFile_GetResForkSize	get the size of the resource fork of a file
TrFile_GetTimeCreated	get the creation time for the file specified by the FileSpec
TrFile_GetTimeModified	get the modification time for the file specified by the FileSpec
TrFile_GetTimestampCreated	get the creation date and time of a file
TrFile_GetTimestampModified	get the modification date and time of a file
TrFile_GetTypeOfItem	get the type of an item, for example if it is a folder or a file
TrFile_GetZipInfo	get information from a ZIP file
TrFile_InsertContents	insert text into a file at a specified position
TrFile_Launch	opens a file with the file's application

Summary of functions (continued)

function name	short description
TrFile_ListDisks	lists the names of all currently available disks
TrFile_ListFolder	lists the contents of a folder
TrFile_MetaData	gets metadata, out of an image file
TrFile_MountDisk	connect to a shared network volume, by mounting the disk
TrFile_MoveFile	moves (or renames) a file from one disk location to another
TrFile_MoveFolder	moves (or renames) a folder to the specified folder path.
TrFile_Reveal	shows a file (or folder) in its enclosing folder
TrFile_SaveFileDialog	presents the user with a save file dialog
TrFile_Search	searches a volume (disk) for a file or folder (directory)
TrFile_SelectFileDialog	presents the user with a standard dialog and displays all files in a directory
TrFile_SelectFolderDialog	shows the user a folder selection dialog
TrFile_SetContents	sets the contents of an existing file
TrFile_SetDefaultCreator	specifies the file creator to be used when creating a new file
TrFile_SetDefaultFileSpec	sets the FileSpec (i.e. the file) to be used in succeeding functions
TrFile_SetDefaultType	sets the FileType for displaying and creating new files
TrFile_SetFileAttribute	sets an attribute of the file specified by the FileSpec
TrFile_SetMetaData	sets metadata into an image file (JPEG)
TrFile_SetTimestampCreated	sets the creation date/time of the file specified by the FileSpec.
TrFile_SetTimestampModified	sets the modification date/time of the file specified by the FileSpec.
TrFile_Substitute	substitutes characters in a (text) file
TrFile_UnmountDisk	unmounts and ejects a remote or removable disk
TrFile_UnZip	expands a ZIP file into the specified folder
TrFile_Version	use this function to see which version of the plug-in is loaded
TrFile_VersionAutoUpdate	standard version number for AutoUpdate of FileMaker Server

Function Reference

TrFile_AppendContents

Syntax TrFile_AppendContents(switches; text)

Append characters to the contents of an existing file indicated by the SetDefaultFileSpec function.

Parameters

switches	these alter the behavior of the function
text	the characters that you want to add at the end of the default file

Switches can be empty or one of this:

-Encoding=Native	(default)
-Encoding=UTF8	
-Encoding=ASCII_DOS	
-Encoding=ASCII_Windows	
-Encoding=ASCII_Mac	
-Encoding=ISO_8859_1	(Windows Latin-1)
-Encoding=BytesOnly	only characters in the range 0-255 are written, others are skipped

This determines the character encoding of the appended data.

You can also add this switch:
-CRtoCRLF Use Windows line endings. Returns in the text parameter are substituted with CRLF (Carriage Return followed by a Line Feed) in the written file.

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The string of characters were added to the file
$$-1	genericErr	The file could not be opened for appending (You might need to create the file first).
$$-2	genericErr	The file could not be opened for appending (You might need to create the file first).

Other errors may be returned.

Special considerations

Use the function TrFile_SetDefaultFileSpec first to set the default file. See also the functions:
TrFile_Save FileSpec Dialog to ask the user for a FileSpec, TrFile_SetDefaultFileSpec to indicate the file to affect.

Note that the maximum size of TrFile_AppendContents is currently 150 million characters (per call to the function). This equals to about max. 600 million characters written. This depends on the encoding, for example one Unicode character encoded as UTF-8 can require up to 4 character in the written file.

Up to v5.0 there was an error in the documentation: it was incorrectly stated that UTF-8 was the default encoding. The default is -Encoding=Native, which is the current encoding as set in the system preference, for example on western Windows systems it usually is ISO_8859_1 or ASCII_Windows. On Mac OS X the default will be ASCII_Mac.
To be able to export all possible characters (like chinese characters) it is best to export with encoding UTF8.

Example usage

Assume your C disk already contains a file "Testtext.txt", which contains the text "Hello". We assume that a global number field gErrorCode is defined. In ScriptMaker create the following script:

```
Set Field[gErrorCode, TrFile_SetDefaultFileSpec( "-Unused" ; "C:\Testtext.txt" )]
If [gErrorCode = 0]
    Set Field[gErrorCode, TrFile_AppendContents( "-Unused" ; " there." )]
```

```
    End If
```

This script will add the text " there." to the end of the file. After running the script once the file will contain "Hello there.". If you run the script again the file will contain "Hello there. there."

Example 2

This shows how to write to a log file. We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number	gXplatformReturn	Global, text
gLogFilePath	Global, text	gLogtext	Global, text

gLogFilePath should contain the path to an existing log file, for example "D:\Logs\L2005_01.TXT" (Windows) or "Mac HD:Logs:Log 2000_01" (Mac). gXplatformReturn should contain a return on Mac and a return and a linefeed on Windows. gLogtext is the text you want to log. Add the following scriptstep to your startup script:

```
Set Field[gErrorCode, TrFile_SetDefaultFileSpec( "-Unused" ; gLogFilePath )]
```

This will set the default file to your log file. Then whenever you need to write a line to the log fill the field gLogtext and perform this script step:

```
Set Field[gErrorCode, TrFile_AppendContents(  "-Encoding=UTF8" ;
        DateToText(Get(CurrentDate)) & " " &
        TimeToText(Get(CurrentTime)) & " text: " & gLogtext & gXplatformReturn))]
```

This script step will add a line every time this script is performed. The log file might look like this:

12/31/2005 10:23:56 text: user Peter logged in
12/31/2005 14:31:02 text: user Peter logged out
12/31/2005 14:31:02 text: closed file "Invoices.fp7"
etc...

TrFile_AsciiValueToText

Syntax TrFile_AsciiValueToText(switches ; ASCIIvalues { ; separator })

Converts (one or more) numbers to their equivalent ASCII characters.

Parameters

switches	these alter the behavior of the function
ASCIIvalues	one or more numbers in the range from 0-255 separated by a separator
separator	(optional) the separator between the values, if you omit this parameter " " and	is used.

Switches can be empty or one of this:

-Encoding=Native	(default) use Unicode encoding for the higher ASCII's 128-255
-Encoding=ASCII_Mac	use Mac ASCII for the higher ASCII's 128-255 (as used in fmp 6)

Returned result

The converted ASCII text

Special considerations

You can also use hexadecimal notation for the numbers. Use 0x00...0xFF to indicate hexadecimal notation.
The graphic rendition of characters greater than 127 is undefined in the American Standard Code for Information Interchange (ASCII Standard) and varies from font to font and from computer to computer and may look different when printed.
Values higher than 255 are ignored.

This function is the same as the Serial_AsciiValueToText in the Troi Serial plug-in.

Example usage

Set Field [text, TrFile_AsciiValueToText ("-Unused" ; "65 65 80 13")]
or
Set Field [text, TrFile_AsciiValueToText ("-Unused" ; "65|65|80|13")]

This will both result in the text "AAP<CR>" where <CR> is a Carriage Return character

Example 2

Set Field [text, TrFile_AsciiValueToText("-Unused" ; "0x31-0x32-0x33-0x0D-0x0A" ; "-")]

This will result in the text "123<CR><LF>" where <CR> is a Carriage Return and <LF> is a Line Feed character.

Example 3

The switch -Encoding=ASCII_Mac is to be able to encode all 256 values possible in FileMaker Pro 6 to their converted values in FileMaker Pro 7 and later. For example the Apple Logo character with Mac ASCII code 240 is converted to Unicode 63743 in FileMaker 7.

Set Field [text, TrFile_AsciiValueToText("-Encoding=ASCII_Mac" ; "240")]

This will result in the text "" .

TrFile_ContentsDialog

Syntax TrFile_ContentsDialog(switches ; { prompt ; { initialFolder } })

Shows the user a file selection dialog, the contents of the selected file is returned.
This function is a combination of "TrFile_Get FileSpec Dialog" followed by a "TrFile_GetContents" function.

Parameters

switches	these alter the behavior of the function
prompt	(optional) adds a prompt text to the GetFile Dialog. It may be left empty
initialFolder	(optional) the FileSpec or path to the folder where the dialog initially starts

Switches can be empty or one of this:
-Encoding=Native (default)
-Encoding=UTF8
-Encoding=ASCII_DOS
-Encoding=ASCII_Windows
-Encoding=ASCII_Mac

This determines the character encoding of the text to be read.

You can also add this switch:
-ConvertToFMPLinebreaks this will convert any line breaks in the data to the line break FileMaker expects: CRLF, LF and CR will all become CR.

Returned result

The characters of the file the user selected. If the user pressed cancel an error code "$$-1" is returned.

Special considerations

You can read up to 2 Gb into a field. However more than 500000 characters will slow down FileMaker considerably.

Only use -ConvertToFMPLinebreaks when getting text files, as it might change the data returned. If for example you use it with an image file the imported image might get corrupted.

On Mac the dialog will filter and show only text files by default. To change the default filtering use the TrFile_SetDefaultType function, or take a look at "Filtering Files" in the example databases.

Up to v5.0 there was an error in the documentation: it was incorrectly stated that UTF-8 was the default encoding. The default is -Encoding=Native, which is the current encoding as set in the system preference, for example on western Windows systems it usually is ISO_8859_1 or ASCII_Windows.

Example usage

```
Set Field [ result, TrFile_ContentsDialog( "-Unused" ;  "Please select the text file you want to be used:|D:\DataFiles\" )]
```

You can also use a calculation for the prompt, for example this one which uses a global text field:

```
Set Field [ result, TrFile_ContentsDialog( "-Unused" ; "Please select the Excel file """ & gFileName & """:" )]
```

This will result in a file selection dialog with this prompt text: 'Please select the Excel file "invoices99.xls":'.

TrFile_Control

Syntax TrFile_Control(switches; { password })

Controls the functions of the plug-in. You can disable and enable the functions of the plug-in. This allows you to create powerful solutions, without the risk of users misusing the functions, causing unwanted or harmful results.

Parameters

switches	these determine how the function works. You can disable or enable all functions
password	the password to be used

switches can be one of the following:

-DisableAllFunctions	allow the user no access to functions
-EnableAllFunctions	allow the user access to all functions
-CurrentEnabledStatus	get the status, 1 = plug-in is enabled, 0 plug-in is disabled

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	
$$-4217	pwdAlreadySet	password already set (enable first)
$$-4218	alreadyEnabled	functions are already enabled
$$-4219	pwdWrong	password was wrong

Other errors may be returned.

Special considerations

When you call a disabled function an error code of $$-4220 is returned.

After restart of FileMaker all the functions of the plug-in are reenabled.

Example usage

```
Set Field[result, TrFile_Control( "-disableAllFunctions"  ; "secret password")]
```

This will disable the functions of the plug-in.

Example 2

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gPassword	Global, text

gPassword should contain a password, for example "rapunsel". Add the following to the startup script:

```
Set Field[gErrorCode, TrFile_Control( "-disableAllFunctions"  ; gPassword)]
```

This will disable the plug-in function. When you want to use the plug-in add the following script steps:

```
Set Field[gErrorCode, TrFile_Control( "-enableAllFunctions" ; gPassword)]
... add powerful functions here...
Set Field[gErrorCode, TrFile_Control( "-disableAllFunctions"  ;  gPassword)]
```

This will temporarily enable the plug-in functions.

TrFile_ConvertFromFMText

Syntax TrFile_ConvertFromFMText(switches ; text)

Converts text formatted in the internal FileMaker format to text formatted in the format of the current operating system running on this computer. On Windows the High ASCII characters (128-255) are not the same as stored internally in FileMaker.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
text	the text in internal FileMaker format

Returned result

The text in the native format of the operating system of this computer. This will be Windows format on Windows and Mac OS format on Mac OS.

Special considerations

NOTE Function not recommended, only for compatibility with older plug-in. Use the "-Encoding=xxx" switches of TrFile_AppendContents and TrFile_SetContents instead. This function also converts line endings. In the internal FileMaker format (and on Mac) lines end with a single <CR> character. On Windows this function will replace all CR's to the normal line ending on Windows, the two characters <CR><LF>.

Example usage

Assume that on Windows we want to create the file "HighText.Doc", which contains the text "Bjørn, <CR><LF>Münchenstraße 2". Add this script step:

```
Set Field[result, TrFile_ConvertFromFMText( "-Unused" ; "Bjørn, ¶Münchenstraße 2")]
```

The result will look like:
"Bj¯rn,

M¸nchenstrafle 2".

This may look wrong, but when you use for example the TrFile_SetContents function it will appear correct in the file.

Example 2

We assume that in your FileMaker file the following fields are defined:

contents	text
gErrorCode	Global, number
gOutputFile	Global, text
gContentsConverted	Global, text

gOutputFile should contain the path to a non-existing file, for example "D:\Out.txt" (Windows) or "Mac HD:Out.txt" (Mac). In a script add the following script steps:

```
Set Field[gContentsConverted, TrFile_ConvertFromFMText( "-Unused" ; contents)]
## now gContentsConverted is in the format of the operating system
Set Field[gErrorCode, TrFile_CreateFile( "-Unused" ; gOutputFile)"]
Set Field[gErrorCode, TrFile_SetDefaultFileSpec( "-Unused" ; gOutputFile)]
If [gErrorCode = 0 ]
   Set Field[gErrorCode, TrFile_SetContents( "-Unused" ; gContentsConverted)]
End If
```

This will get the contents of the contents field into the file . In the first step the high ASCII characters will be converted from the internal FileMaker format. This will work transparent on both Mac OS and Windows.

TrFile_ConvertToFMText

Syntax TrFile_ConvertToFMText(switches ; text)

Converts text formatted in the format of the current operating system running on this computer to text formatted in the internal FileMaker format. On Windows the High ASCII characters (128-255) are not the same as stored internally in FileMaker.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
text	the text in the native format of the operating system of this computer. This will be Windows format on Windows and Mac OS format on Mac OS

Returned result

The text in internal FileMaker format.

Special considerations

NOTE Function not recommended, only for compatbility with older plug-in. Use the "-Encoding=xxx" switches of TrFile_AppendContents and TrFile_SetContents instead. This function also converts line endings to the internal format. On Windows lines end with the two ASCII characters <CR><LF>. This is converted to the single character <CR> as used by the internal FileMaker format.

Example usage

Assume that on Windows we have the file "HighText.Doc", which contains the text "Bjørn, <CR><LF>Münchenstraße 2". Add this script step:

```
Set Field[result, TrFile_ConvertToFMText( "-Unused" ;  TrFile_GetContents( "-Unused" ; "C:\HighText.Doc"))]
```

This will store the contents of the "HighText.Doc" in the result field, with the high ASCII characters converted to the internal FileMaker format. In the result field this will look like this:

```
Bjørn,¶
Münchenstraße 2
```

Example 2

We assume that in your FileMaker file the following fields are defined:

contents	text
gFileSpec	Global, text

gFileSpec should contain the path to an existing file, for example "D:\Readme.txt" (Windows) or "Mac HD:Readme.txt" (Mac). In a script add the following script step:

```
Set Field[contents, TrFile_GetContents( "-Unused" ; gFileSpec)]
## now the contents field is in the format of the operating system
Set Field[contents, TrFile_ConvertToFMText( "-Unused" ; contents)]
```

This will get the contents of the file into the contents field. In the second step the high ASCII characters will be converted to the internal FileMaker format. This will work transparant on both Mac OS and Windows.

TrFile_CopyFile

Syntax TrFile_CopyFile(switches ; sourceFilepath ; destinationFilePath)

Copies a file to the specified path.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
sourceFilePath	the path to the file to copy
destinationFilePath	the path where the file must be copied to

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The file was copied
$$-43	fnfErr	Source file not found, or destination directory does not exist
$$-48	dupFNErr	Destination file already exists
$$-4254	returnInPathErr	The path contains an illegal return character
$$-1	genericErr	The file could not be copied

Other errors may be returned.

Special considerations

NEW you can use FileMaker styled paths now, like "filemac:/MacHD/folderA/test.txt".
Use the TrFile_DeleteFile to delete a file first if it exists. See also the function TrFile_SaveFileDialog to get a FileSpec for a file.

Example usage

Assume your C disk already contains a file "Testtext.txt". We assume that a global number field gErrorCode is defined. Create the following script:

```
Set Field[gErrorCode, TrFile_CopyFile( "-Unused" ; "C:\Testtext.txt" ; "D:\MyFiles\TestDupl.txt" )]
```

This script will copy the file "Testtext.txt" to the "TestDupl.txt" file in directory "MyFiles" on the D: disk. On Mac OS the paths will be of the form "Mac HD:MyFiles:TestDupl.txt".

Example 2

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gSourceFilePath	Global, text
gDestFilePath	Global, text

gSourceFilePath should contain the path to an existing file, for example "D:\Logs\Log01.TXT" (Windows) or "Mac HD:Logs:Log 1" (Mac). gDestFilePath should contain the path to the destination and should not exist, for example "D:\Logs\L2000_01.TXT" (Windows) or "Mac HD:Logs:Log 2000_01" (Mac). In a script add the following script step:

```
Set Field[gErrorCode, TrFile_CopyFile( "-Unused" ; gSourceFilePath ; gDestFilePath )]
```

This will copy the source file to the path indicated in the gDestFilePath.

TrFile_CopyFolder

Syntax TrFile_CopyFolder(switches ; sourceFolderPath ; destinationFolderPath)

Copies a folder to the specified folder path.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
sourceFolderPath	the path to the folder to copy
destinationFolderPath	the path where the folder must be copied to

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The folder was copied
$$-43	fnfErr	Source folder not found, or parent folder of destination folder does not exist
$$-48	dupFNErr	Destination folder already exists
$$-1234		The destination is inside the source
$$-1407		The source is not a folder

Other errors may be returned.

Special considerations

NEW you can use FileMaker styled paths now, like "filewin:/C:/dir/subdir".
Use the TrFile_DeleteFolder to delete the destination folder first if it exists.

Example usage

Assume your C disk already contains a folder "TestFolder" and a folder "MyFiles". We assume that a global number field gErrorCode is defined. Create the following script:

```
Set Field[gErrorCode, TrFile_CopyFolder( "-Unused" ; "C:\TestFolder" ; "C:\MyFiles\CopiedFolder" )]
```

This script will copy the folder"TestFolder" and its contents to the "CopiedFolder" file inside the folder "MyFiles" on the C: disk. On Mac OS X the paths will be of the form "Mac HD:MyFiles:CopiedFolder".
Notice that the destination folder has been renamed too.

Example 2

We assume that in your FileMaker file the following fields are defined:

gSourceFolderPath	Global, text	gErrorCode	Global, number
gDestFolderPath	Global, text		

gSourceFolderPath should contain the path to an existing folder, for example "D:\Logs\" (Windows) or "Mac HD:Logs:" (Mac). gDestFolderPath should contain the path to the destination and should not exist, for example "D:\Logs\L2007_12" (Windows) or "Mac HD:Logs:Log 2007_12" (Mac). In a script add the following script step:

```
Set Field[gErrorCode, TrFile_CopyFile( "-Unused" ; gSourceFolderPath ;  gDestFolderPath )]
```

This will copy the source folder to the path indicated in the gDestFolderPath.

TrFile_CreateAlias

Syntax TrFile_CreateAlias(switches ; sourcePath ; destAliasSpec)

Create an alias file (or shortcut) from the source file or source folder.

Parameters

switches	modifies the behavior of the function
sourcePath	the path to the file, folder or disk of which an alias (shortcut) must be created
destAliasSpec	the path to the alias file

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The ZIP file was unzipped
$$-43	fnfErr	Source file/folder not found
$$-48	dupFNErr	Destination alias already exists

Other errors may be returned.

Special considerations

To make it easier to find a file, folder, app, or disk, you can create an alias for it in an easy-to-find location. When you open an alias in the Finder or Windows explorer, the original item opens.
On Windows this is called an shortcut, and always has the extension .lnk.

Example usage

Set Field[$ErrorCode ; TrFile_CreateAlias("-Unused" ; "D:\myReport2016.pdf"; "C:\workDir\currentReport.pdf")]

This will create the shortcut currentReport.pdf.lnk in the folder workDir, which links to the original myReport2016.pdf file.

Example 2

We assume that in your FileMaker file the following fields are defined and filled with the paths to the source file (or folder) and destination alias:

this::sourcePath may contain something like "MacHD:Users:kip:Report folder"
this::destAlias_Path may contain something like "MacHD:Users:Desktop:AliasToFolder"

This example will first set the variables from the paths stored in the two fields, and then expand the ZIP file. In a script add the following steps:

```
#Initialize: copy the fields into variables first:
Set Variable [ $SourcePath;  this::sourcePath ]
Set Variable [ $DestAlias;  this::destAlias_Path ]

#Now create the alias:
Set Variable [ $ErrorCode;  TrFile_CreateAlias( "-Unused"  ; $SourcePath ; $DestAlias) ]

#Return the resulting error code:
Exit Script [ $ErrorCode ]
```

This will create the alias based on the field contents. With the sample data the alias AliasToFolder will be created on the desktop and links back to the Report Folder.

TrFile_CreateFile

Syntax TrFile_CreateFile(switches ; filePath)

Creates a new empty file in the location indicated by the filePath. This function requires no user intervention.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
filePath	the path to the file to create

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The file was created
$$-48	dupFNErr	Destination file already exists
$$-1	genericErr	The file could not created

Other errors may be returned.

Special considerations

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".
See also the function TrFile_SaveFileDialog to get a filepath for the file.

Example usage

Set Field[gErrorCode, TrFile_CreateFile("-Unused" ; "C:\Testtext.txt")]

This will create the empty file "Testtext.txt" on the C disk.

Example 2

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gDestFilePath	Global, text

gDestFilePath should contain the path to the destination and should not exist, for example "D:\Logs\L2000_01.TXT" (Windows) or "Mac HD:Logs:Log 2000_01" (Mac). In a script add the following scriptstep:

```
Set Field[gErrorCode, TrFile_CreateFile( "-Unused" ; gDestFilePath )]
```

This will create the file indicated in the gDestFilePath.

TrFile_CreateFolder

Syntax TrFile_CreateFolder(switches ; folderPath)

Creates a new (empty) folder (subdirectory).

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
folderPath	the path to the folder to create

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The folder was created
$$-48	dupFNErr	Destination folder already exists
$$-1	genericErr	The folder could not be created
$$-50	paramErr	Parameter error

Other errors may be returned.

Special considerations

NEW you can use FileMaker styled paths now, like "filemac:/MacHD/folder/subfolder".
See also the functions: TrFile_SaveFileDialog to get a folderPath for a folder, and TrFile_CreateFile to create a new empty file.

v6.0 improved the TrFile_CreateFolder function: you can now specify a path and the plug-in will create the folder, including all folders in the path that do not exist.

Example usage

We assume that a global number field gErrorCode is defined. Create the following script:

```
Set Field[gErrorCode, TrFile_CreateFolder( "-Unused" ; "C:\Data\NewFolder" )]
```

This script will create the folder "NewFolder" inside the Data folder on the C disk.
On Mac OS this will be:

```
Set Field[gErrorCode, TrFile_CreateFolder( "-Unused" ;  "MyDisk:Data:NewFolder" )]
```

Example 2

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gFolderPath	Global, text

gFolderPath should contain the path to the folder to be created and should not exist, for example "D:\Logs\Data" (Windows) or "Mac HD:Logs:Data" (Mac OS). In a script add the following scriptstep:

```
Set Field[gErrorCode, TrFile_CreateFolder( "-Unused" ; gFolderPath )]
```

This will create the folder specified in the gFolderPath.

TrFile_CreateThumbnail

Syntax TrFile_CreateThumbnail(switches ; FileSpec ; {thumbnailName})

Creates a thumbnail (a small image of normally 80x80 pixels) from a graphics file and returns it.

Parameters

switches these determine the properties of the thumbnail that is returned
FilePath the path to the file from which to create a thumbnail
thumbnailName (optional) the (internal) name of the created thumbnail

Switches can be empty or one or more of the following:
-NoDialog don't show a progress dialog
-Size=80 (default) maximum size of the height or width is 80 pixels
-Size=128 maximum size of the height or width is 128 pixels
-Size=256 maximum size of the height or width is 256 pixels, more like a preview =)
-BitDepth=8 use 8 bits (=256 colors) instead of 16 bits (=65536 colors)
-BitDepth=32 use 32 bits instead of 16 bits

You can also add these switches:
-AllowAllThumbnailSizes this will make it possible to use ANY size for the -Size switch. Use with care
-AntiAliasThumbnail creates bicubic antialiased thumbnails

-Square create square thumbnails (the extending parts of the wider side of the image are cropped)
-IgnoreAlphaChannel ignore the alpha channel (for transparency), treat it as completely opaque when creating the thumbnail.
-CreatePNG creates a PNG thumbnail, with an alpha channel. It has the same transparent areas as the original.

Returned result

The returned result is the thumbnail itself. If an error occurs an error code is returned. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

$$-2003	cantFindHandler	No thumbnail could be created

Other errors may be returned.

Special considerations

The "-Square" switch will create square thumbnails. In this case the thumbnails will have the extending parts of the wider side of the image cropped (thus showing the middle square of the image). Also note that the image will be scaled proportionally up to a square, if one of the sides of the image is smaller than the requested size. This makes sure the result is always a square with the exact dimensions requested.

The optional thumbnailName is added to the created thumbnail image data and also when the thumbnail is stored in the container. This name is normally not visible, but will be used for example when you perform a Export Field Contents action later.

You can also use FileMaker styled paths, like "imagemac:/MacHD/folder/myImage.jpg".
An error code $$-2003 is returned when you try to create a thumbnail for a file that has no graphics, for example a text file or a spreadsheet. It is no problem if you don't know if a file has a thumbnail. Just try to create a thumbnail, if you get the error $$-2003 you know it is not possible.
NOTE QuickTime is no longer required for this function.

Example usage

```
Set Field [ container field, TrFile_CreateThumbnail( "-Unused" ; "Mac HD:photo.jpeg" )]
```

This will create a thumbnail and put it in a container field.

TrFile_CreateThumbnail

Example 2

Say you want to create the thumbnails of filepaths that are stored in a set of records. We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
filePath	text, contains the path to a file
thumbnailField	container

filePath should contain the full path to existing files, for example "Mac HD:Aap.JPG" (Mac). In a script add the following script steps:

```
Go to Record[first]
Loop
    Set Field[thumbnailField , TrFile_CreateThumbnail( "-Unused" ; filePath )]
    Set Field[gErrorCode , GetAsText(thumbnailField)]
    Exit Loop If[Left(gErrorCode  ; 2)  = "$$") ]
    Go to Record[next, exit after last]
End Loop
```

This will try to create thumbnails for all files for the records of the found set.

TrFile_CreateZip

Syntax TrFile_CreateZip(switches ; sourcePath ; destZipFile)

create a ZIP file, from the specified file or folder.

Parameters

switches	this alters the behavior of the function
sourcePath	the path to the source file or folder to be zipped
destZipFile	the path to the new ZIP file to be created

Switches can be empty or this:

-TimeoutSecs=x	specify the timeout time in x seconds (default = 20 seconds)

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The ZIP file was created
$$-43	fnfErr	Source file not found
$$-48	dupFNErr	Destination ZIP file already exists

Other errors may be returned.

Special considerations

You can zip a single file or a complete folder, including subfolders and all items contained in the folder.
Zipping much items can take a long time. With the "-TimeoutSecs=x" switch you can increase the timeout time so it is long enough.

Example usage

Set Field[$ErrorCode; TrFile_CreateZip("-Unused" ; "C:\dataDir\" ; "D:\myZIP2015.zip")]

This will create a compressed ZIP file archive myZIP2015.zip with the contents of the folder dataDir.

Example 2

We assume that in your FileMaker file the following fields are defined and filled with the paths to the source and destination ZIP file:

this::sourcePath	may contain something like "MacHD:Users:kip:bigtext.doc"
this::destinationZIPPath	may contain something like "MacHD:Users:kip:bigtext.zip"

This example will first set the variables from the paths stored in the two fields, and then create the ZIP file:
In a script add the following steps:

```
#Initialize: copy the fields into variables first:
Set Variable [ $SourcePath; this::sourcePath ]
Set Variable [ $DestinationZIP; this::destinationZIPPath ]
#
#Now zip it:
Set Variable [ $ErrorCode; TrFile_CreateZip( "-Unused" ; $SourcePath ; $DestinationZIP) ]
```

TrFile_DeleteFile

Syntax TrFile_DeleteFile(switches ; filePath)

Deletes the file indicated by the FileSpec. This function requires no user intervention.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
filePath	the path to the file to delete

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The file was deleted
$$-43	fnfErr	The file was not found
$$-47	fBsyErr	The file is in use
$$-1	genericErr	The file could not be deleted

Other errors may be returned.

Special considerations

WARNING: The file is not moved to the Trash, but deleted completely. This can not be undone!
See also the Dial_Dialog function (of the Troi Dialog plug-in) or use the "Show Message" script step if you want to warn the user.

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

Assume your C disk already contains a file "TestFile.xls". We assume that a global number field gErrorCode is defined. Create the following script:

```
Set Field[gErrorCode, TrFile_DeleteFile( "-Unused" ; "C:\TestFile.xls" )]
```

This script will delete the file "TestFile.xls". On Mac OS the path will be of the form "Mac HD:TestFile.xls".

Example 2

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gFilePath	Global, text

gFilePath should contain the path to an existing file which you want to delete, for example "D:\Logs\Log01.TXT" (Windows) or "Mac HD:Logs:Log 1" (Mac). In a script add the following scriptstep:

```
Set Field[gErrorCode, TrFile_DeleteFile( "-Unused" ;  gFilePath )]
```

This will delete the file from the disk.

TrFile_DeleteFolder

Syntax TrFile_DeleteFolder(switches ; folderPath)

Deletes the folder indicated by the folderPath.

Parameters

switches	determines which folders are deleted
folderPath	the folder path to the folder to delete

Switches can be empty or this:

-DeleteAllSubFoldersAndAllContents	delete all sub folders and also the contents of those folders.

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The folder was deleted
$$-1		user CANCELED
$$-37	bdNamErr	Bad name in the file system
$$-47	fBsyErr	File is busy, can't delete because it is not empty
$$-50	paramErr	Parameter error, check if the supplied parameters are correct.
$$-120	dirNFErr	Directory not found,is not a directory or is a file

Other errors may be returned.

Special considerations

WARNING This is a powerful feature. Be careful what you do! Note also that the folder is not moved to the Trash, but deleted completely. This can not be undone!
See also Dial_Dialog function (of the Troi Dialog plug-in) or use the "Show Message" script step if you want to warn the user.
NEW you can use FileMaker styled paths, like "filemac:/MacHD/folder/subfolder".

Example usage

Assume your C disk already contains a folder "TestFold". We assume that a global number field gErrorCode is defined. Create the following script:

```
Set Field[result, TrFile_DeleteFolder( "-Unused " ; " C:\TestFold\" )]
```

This script will delete the folder "TestFold". On Mac OS the path will be of the form "Mac HD:TestFold:".

Example 2

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gFolderPath	Global, text

gFolderPath should contain the path to an existing file which you want to delete, for example "D:\Logs\" (Windows) or "Mac HD:Logs:" (Mac). In a script add the following scriptstep:

```
Set Field[gErrorCode, TrFile_DeleteFolder( "-DeleteAllSubFoldersAndAllContents" ;  gFolderPath )]
```

This will delete the folder from the disk, including contents, like files and folders. The user is always given a warning. If you don't use this switch only empty folders are deleted.

TrFile_DiskInfo

Syntax TrFile_DiskInfo(switches ; diskname)

Retrieves information about a specified disk, like the number of files.

Parameters

switches	this determines which information is returned
diskname	the name of the disk

Switches are different for each platform. For all platforms the switch can be one of this:

-GetIsNetworkDisk	returns 1 if it is a remote network disk or 0 if not.
-GetFreeMegs	get the number of free Mega bytes (Mb) on this disk
-GetTotalMegs	get the size of this disk in Mega bytes (Mb)

Switches are different for each platform. For Mac OS the switch can be one of this:

-GetFileCount	get the number of files on this disk
-GetFolderCount	get the number of folders on this disk

For Windows the switch can be one of this:

-GetVolumeLabel	get the label of the disk, for example "C Disk"
-GetVolumeSerialNumber	get the serialnumber of the disk, for example "123234"
-GetUniversalName	get the Universal Name of a shared network drive
-GetDriveLetter	get the drive letter from an Universal Name of a shared network drive

Returned result

The returned result is a number or an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

$$-35	nsvErr	No such volume, disk not found
$$-50	paramErr	Parameter error, check if the supplied parameters are correct
$$2250		Not connected to a network

Other errors may be returned.

Special considerations

The switches -GetUniversalName and -GetDriveLetter are available in the 2.5x2 version or later on Windows only.

Example usage

Set Field[result, TrFile_DiskInfo(" -GetTotalMegs" ; "C:")]
Set Field[result, TrFile_DiskInfo(" -GetFreeMegs" ; "Server1")]
Set Field[result, TrFile_DiskInfo(" -GetFileCount " ; "Mac HD")]

These steps will return the total and the free number of Mb of the disk. The last step will return the number of files for of the disk "Mac HD". Note this is available on Mac OS only.

Set Field[result , TrFile_DiskInfo(" -GetVolumeSerialNumber " ; "C:)]

This will return the serialnumber of the hard disk C. Note this is available on Windows only.

Example 2

We assume that in your FileMaker file the following fields are defined:

gFolderCount	Global, number
gDiskName	Global, text

gDiskName should contain the name of a hard disk. In a script add the following script step:

Set Field[gFolderCount, TrFile_DiskInfo(" -GetFolderCount " ; " _gDiskName)]

This will return the number of folders on the disk.

Example 3

This example describes how to work with UNC's, which are only available on Windows. We assume that in your FileMaker file the following fields are defined:

gFolderCount	Global, number
gDiskName	Global, text
gUNC_Name	Global, text

gDiskName should contain the name of a remote hard disk, for example "Z:". In a script add the following script steps:

Set Field[gUNC_Name, TrFile_DiskInfo(" -GetUniversalName " ; gDiskName)]

This will return the UNC, for example "\\Server3\\SharedDocs". If you want do the reverse add this step:

Set Field[gDiskName, TrFile_DiskInfo(" -GetDriveLetter " ; " gUNC_Name)]

To mount a shared hard disk with a UNC see the function TrFile_MountDisk.

TrFile_DragAndDrop

Syntax TrFile_DragAndDrop(switches ; fileName ; scriptName ; { fieldBounds })

This function implements drag and drop of files and folders onto a FileMaker window.

Parameters

switches determine the behavior of the function
fileName the name of the file that contains the script to be triggered
scriptName the name of the script to be triggered when files/folders are dropped on a window
fieldBounds (optional) a rectangle on the layout where the drag is accepted. Dimensions (in pixels) are in this order: "left top right bottom", which is the same order as the FileMaker FieldBounds function returns

switches can be one of this:
-AddDragAndDropHandler the plug-in starts accepting drag and drop. You can now drag files on the window
-StopDragAndDrop the plug-in no longer accepts drag and drop

You can also add one of these switches
-LinkCursor (default) while dragging show an arrow cursor with a small link symbol
-CopyCursor while dragging show arrow cursor with a small plus symbol

These 2 switches only change how the cursor looks. In the triggered script you need to implement any action you want to perform on the dragged items.

You can now also add:
-AllowMailDrops this will allow emails (from the OS X Mail.app) to be dragged to FileMaker

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	
$$-50	paramErr	there was an error with a parameter
$$-5600	errInvalidWindowRef	the window could not be found
$$-4223	errTooManyEvents	you have specified more than 2 D&D handlers

Other errors may be returned.

Special considerations

You can now specify two separate drag destinations, by calling the function twice with the switch -AddDragAndDropHandler with the 2 rectangles you want as drop zones.

The FileMaker window that is in front when the "-AddDragAndDropHandler" switch is executed, will be activated as the destination of future drags.

When the plug-in detects a drag on the destination window, the specified script will be triggered. The (incoming) ScriptParameter of the triggered script will contain the list of files/folders that were dragged on the window.

You can drag one or more files and folders (or a mix of this) on the window. The trigger script should be expecting this and be capable to handle this, for example in a loop. Alternatively the script can only use the first file in the list.

See the new switch "-PackageFolderAttr" in the function TrFile_GetFileAttribute: this can be used to determine if a result of the drag and drop action is a package.

KNOWN ISSUES
- The plug-in is not aware if the position of the custom fieldbounds is changed, for example when scrolled.
- (Mac OS X) the drop box highlight is only visible over a white background.
- (Mac OS X) the link and copy cursor is visible only when moving the cursor.

TrFile_DragAndDrop

Example usage

To make drag and drop work you need a trigger script that handles the drag. Also you need to start and stop the plug-in handling the drags. We here create a (simplified) script that will put the path of first file found in a field. See the example files for a more extensive script handling more files and container fields.

- Create a script named "DragTriggerscript":

```
Set Variable [ $pathList; Value:Get ( ScriptParameter ) ]
If [ Get ( FoundCount ) = 0 ]
    New Record/Request
End If
Set Variable [ $returnPos; Value:Position($pathList; "¶" ; 1 ; 1 ) ]
Set Field [ this::onepath; Value:If($returnPos > 0 ; Left($pathList ; $returnPos - 1) ; $pathList ) ]
## here you can add actions you want to perform on this file
```

- Starting drag and drop can be done like this:

```
Set Field[gErrorCode, TrFile_DragAndDrop( "-AddDragAndDropHandler " ; Get(FileName) ;  "DragTriggerscript" ))]
```

After this step has been executed the user can drag files on the window and the DragTriggerScript will be run.

- Stopping drag and drop can be done like this:

```
Set Field[gErrorCode, TrFile_DragAndDrop( "-StopDragAndDrop " ; "" ; "" ))]
```

Example 2

```
Set Variable[ $containerFieldBounds, FieldBounds ( Get(FileName) ; Get (LayoutName) ; Get (ActiveFieldName) )]
Set Variable[ $result, TrFile_DragAndDrop( "-AddDragAndDropHandler" ; Get(FileName) ; "DragTriggerscript" ;
$containerFieldBounds)]
```

This will make the drag area around a container field. It works in FileMaker 10. In previous versions of FileMaker you need to correct for the status area on the left. See the scripts in the example file DragAndDrop for this.

TrFile_ExecuteShell

Syntax TrFile_ExecuteShell(switches ; shellCommand)

Execute a command in the (UNIX or Windows) shell of the operating system.

Parameters

switches	these alter the behavior of the function
shellCommand	the command to be executed

Switches can be empty or one of this:

-TimeoutSecs=x	specify the timeout time in x seconds (default = 20 seconds)
-DontWaitOnResult	execute the shell command without waiting on a result
-ShowCommandWindow	(Windows only) show the command window where the shell is running
-WaitAfterExitTicks=x	(Windows only) wait x ticks (= 1/60th of a sec) after the command finishes
-Encoding=Native	(default) uses the native encoding for the command
-Encoding=ASCII_DOS	use OEM DOS ASCII for the higher ASCII's 128-255
-Encoding=ASCII_Windows	use Ansi Windows ASCII for the higher ASCII's 128-255
-Encoding=ASCII_Mac	use Mac ASCII for the higher ASCII's 128-255 (as used in fmp 6)

Returned result

The result of the shell command or an errorcode

If an error occurs it will start with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

$$-4218 kErrCanNotEnable can not start a shell.
$$-4230 timeout error.

Other errors may be returned.

Special considerations

This is a powerful function, with which you can accomplish multiple commands. Please read about the shell command in Unix or Windows before using this.

For Windows: to be able to use spaces and quotes in the command wrap the whole command in 2 extra double quotes, one at the beginning and one at the end, like ""C:\testapp.exe" "test param""

The maximum number for the -TimeoutSecs switch is 1800

Example usage

On Windows:

```
Set Field[gResult, TrFile_ExecuteShell( "-Unused" ; "dir C:\")]
```

On Mac OS X:

```
Set Field[gResult, TrFile_ExecuteShell( "-Unused" ; "ls /")]
```

This will list the contents of the root folder.

Example 2

On Windows:

```
Set Field[gResult, TrFile_ExecuteShell( "-TimeoutSecs=5" ; "echo testtext > C:\example.txt")]
```

TrFile_ExecuteShell

On Mac OS X:

```
Set Field[gResult, TrFile_ExecuteShell( "-TimeoutSecs=5" ; "echo testtext > /example.txt")]
```

This will create a text file example.txt on your startup disk. If the command takes more time than 5 seconds the plug-in will return with an timeout error $$-4230.

TrFile_Exists

Syntax TrFile_Exists(switches ; filePath)

Check for the existence of a file or folder.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want to know if it exists

Returned result

The returned result is a boolean: The function returns 1 if the file or folder exists. If it does not exist or if an error occurs it will return 0.

Special considerations

If the function returns 0 you should decide if it is nessarry to check for errors, for example if a file is on a disk that is not mounted.

Example usage

Set Field[doesExists, TrFile_Exists("-Unused" ; "C:\Test.txt")]

This will return 1 if it exists.

Example 2

We assume that in your FileMaker file the following field is defined:

gFilePath	Global, text

gFilePath should contain the path to a file, for example "D:\Logs\L2000_01.TXT" (Windows) or "Mac HD:Logs:Log 2000_01" (Mac). In ScriptMaker add the following script steps:

```
if [TrFile_Exists( "-Unused" ;  gFilePath )]
  #... do your stuff here
else
  #... do your error handling here....
endif
```

This will make it easy to handle existing files.

TrFile_FileSpecToFullPath

Syntax TrFile_FileSpecToFullPath(switches ; FileSpec)

Returns full path of a file or folder from the FSSpec. Mac OS only.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
FileSpec	the FSSpec of a file or folder. It is of the form ":volumeID:directoryID:fileName"

Returned result

The function TrFile_FileSpecToFullPath is obsolete and now returns error code $$-4266 (kErrNoLongerSupported).

Special considerations

As FSSpecs are deprecated this function no longer is useful.

Example usage

Set Field[result, TrFile_FileSpecToFullPath("-Unused" ; ":-1:300:testfile")]

If the FSSpec is valid this may return something like this:
"Mac HD:Data Files:Database:testfile"

Example 2

We assume that in your FileMaker file the following field is defined:

gFilePath	Global, text

gFilePath should contain the path to an existing file, for example ":-1:300:testfile". In ScriptMaker add the following scriptstep:

Set Field[gFilePath, TrFile_FileSpecToFullPath("-Unused" ; gFilePath)]

This will convert the FSSpec in gFilePath to a Full Path.

TrFile_FindFolder

Syntax TrFile_FindFolder(folderSwitch)

Returns the path to special folders (subdirectories).

Parameters

folderSwitch specifies the name of the wanted special folder.

Not all special folders are available for each platform. On both Mac OS and Windows you can specify:
-Desktop = desktop folder
-System = system folder
-Temporary = hidden temporary folder on the startup disk
-Root = top folder on the startup disk
-Startup = startup items folder in the system folder

On Mac OS you can also specify:
-Applemenu = apple menu folder in the system folder
-Controlpanels = control panels folder in the system folder
-Extensions = extensions folder in the system folder
-Preferences = preferences folder in the system folder
-Shutdown = shutdown items folder in the system folder
-Trash = trash folder on the desktop

On Windows you can also specify:
-Fonts = the fonts folder
-Mydocuments = the My Documents folder
-Programs = the programs folder inside the start menu
-Startmenu = the Startup items folder
-Windows = the Windows folder

On Mac OS X you can also add:
-ReturnFSSpec = the function returns the folder as a FSSpec instead of a full path.

Returned result

FileSpec The Filespec or path to the folder.

If a special folder is not available for that platform $$-1 is returned.

Special considerations

Not every special folder is available for each platform.

Note that there is a trash folder for every volume. Currently this function returns the trash folder on the startup disk.

Some folder are no longer available on Mac OS X, and will return $$-43 (file not found).

Example usage

Set Field [result, TrFile_FindFolder("-Trash")]

This will return the full path of the Trash folder.

TIP: With this folder and the MoveFile function you can move items into the Trash. Be aware that each disk has its one trash, so there may be more than one trash. FindFolder returns the trash of the startup disk.

Example 2

TrFile_FindFolder

Set Field [gStartupDiskFolder, TrFile_FindFolder("-Root")]
This will return the top folder on the startup disk, for example "C:".

TrFile_FullPathToFileSpec

Syntax TrFile_FullPathToFileSpec(switches ; FileSpec)

Changes the full name path to the standard Macintosh FSSpec.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
FileSpec	the Full path of a file or folder. It is of the form "DiskName:foldername1:foldername2:...:foldernameN:filename"

Returned result

The function TrFile_FullPathToFileSpec is obsolete and now returns error code $$-4266 (kErrNoLongerSupported).

Special considerations

As FSSpecs are deprecated this function no longer is useful.

Example usage

Set Field[result, TrFile_FullPathToFileSpec("-Unused" ; "Mac HD:Data Files:Database:testfile")]

If the path is valid this may return something like this:
":-1:300:testfile"

Example 2

We assume that in your FileMaker file the following field is defined:

gFilePath	Global, text

gFilePath should contain the path to an existing file, for example "Mac HD:Data Files:Database:testfile". In a script add the following scriptstep:

```
Set Field[gFilePath, TrFile_FullPathToFileSpec( "-Unused" ; gFilePath )]
```

This will convert the Full Path in gFilePath to an FSSpec.

TrFile_GetContents

Syntax TrFile_GetContents(switches ; filePath ; { start ; { size }})

Returns (a part of) the contents of the file specified by the filePath. You can specify the starting position and the number of characters to get.

Parameters

switches	these alter the behavior of the function
filePath	the path to the file
start	(optional) starting position in the file (with a value of 0 returning the first character)
size	(optional) the number of characters to get

Switches can be empty or one of this:
-Encoding=Native (default)
-Encoding=UTF8
-Encoding=ASCII_DOS
-Encoding=ASCII_Windows
-Encoding=ASCII_Mac

This determines the character encoding of the text to be read.

You can also add this switch:
-ConvertToFMPLinebreaks this will convert any line breaks in the data to the line break FileMaker expects: CRLF, LF and CR will all become CR.

Returned result

The characters of the file specified.

If unsuccessful this function returns an error code starting with $$ and the error code. Possible codes are:
$$-50 = parameter error.
$$-41 = not enough memory
$$-39 = (eofErr) start is beyond the end of the file
$$-4241 = start or size parameter too big (> 2 Gb)

Other error codes might be returned.

Special considerations

The offset of the "start" parameter is from 0. For example a value of 1 will return the contents from the 2nd character.

You can read up to 2 Gb into a field. However more than 500000 characters will slow down FileMaker considerably.

Only use -ConvertToFMPLinebreaks when getting text files, as it might change the data returned. If for example you use it with an image file the imported image might get corrupted.

Up to v5.0 there was an error in the documentation: it was incorrectly stated that UTF-8 was the default encoding. The default is -Encoding=Native, which is the current encoding as set in the system preference, for example on western Windows systems it usually is ISO_8859_1 or ASCII_Windows.

Example usage

Set Field[MyTextField, TrFile_GetContents("-Unused" ; "HD Mac:Text files:My Letter")]
returns the contents of the file.

Example 2

We assume that in your FileMaker file the following fields are defined:

gContents	Global, text	gStart	Global, number
gFileSpec	Global, text	gLength	Global, number

TrFile_GetContents

gFileSpec should contain the path to an existing file, for example "D:\Readme.txt" (Windows) or "Mac HD:Readme.txt" (Mac). In gStart should be the start position and in gLength the number of bytes you want to get. In a script add the following scriptstep:

```
Set Field[gContents, TrFile_GetContents(  "-Unused" ; gFileSpec  ;gStart ; gLength)]
```

This will get part of the contents of the file into the gContents field.

TrFile_GetDataSize

Syntax TrFile_GetDataSize(switches ; filePath)

Returns the size of the data for the file specified by the filePath. The size indicates the actual number of bytes used by the data fork.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file

Returned result

The size (in bytes) of the data of the file specified. On Mac a file can have a data fork and also a resource fork. On Mac OS this function returns the size of the data fork only.

Special considerations

See also the functions TrFile_GetFileSize and TrFile_GetResForkSize.

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

Assume there is a file "Read me" with the text "Hello!" in it. Then:

```
Set Field[MyTextField, TrFile_GetDataSize( "-Unused" ; "Disk:Read me")]
```

returns 6, which is the size of the data in the file. The total file size may be bigger.

Example 2

We assume that in your FileMaker file the following fields are defined:

gFileSpec	Global, text
gSize	Global, number

gFileSpec should contain the path to an existing file, for example "D:\Readme.txt" (Windows) or "Mac HD:Readme.txt" (Mac). In a script add the following scriptstep:

```
Set Field[gSize, TrFile_GetDataSize( "-Unused" ; gFileSpec)]
```

This will fill gSize with the size of the data.

TrFile_GetDateCreated

Syntax TrFile_GetDateCreated(switches ; filePath)

Returns the creation date for the file specified by the filePath.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want the information

Returned result

The returned result is the creation date of the file. The date is in the same format as a date field.

An error code might also be returned. An error always starts with 2 dollars, followed by the error code. Returned error codes can be:

$$-43	fnfErr	File not found, check if the path is valid
$$-1	genericErr	The file could not be found (older versions of the plug-in)

Other errors may be returned.

Special considerations

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

Set Field[creationDate, TrFile_GetDateCreated("-Unused" ; "C:\Test.txt")]

This may return the number 730172 which is equivalent to the date: 22 Feb. 2000.

Example 2

We assume that in your FileMaker file the following fields are defined:

creationDate	date
gFilePath	Global, text

gFilePath should contain the path to the file, for example "D:\Logs\L2000_01.TXT" (Windows) or "Mac HD:Logs:Log 2000_01" (Mac). In a script add the following script step:

```
Set Field[creationDate, TrFile_GetDateCreated( "-Unused" ; gFilePath )]
```

This will store the creation date of the file specified by gFilePath in the field creationDate.

TrFile_GetDateLastAccessed

Syntax TrFile_GetDateLastAccessed(switches ; filePath)

Returns the date a file was accessed (opened) for the last time.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want the information

Returned result

The returned result is the date the file was last accessed. The date is in the same format as a date field.

An error code might also be returned. An error always starts with 2 dollars, followed by the error code. Returned error codes can be:

$$-43	fnfErr	File not found, check if the path is valid

Other errors may be returned.

Special considerations

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

Set Field[lastAccessedDate, TrFile_GetDateLastAccessed("-Unused" ; "C:\Test.txt")]

This may return the number 730172 which is equivalent to the date: 22 Feb. 2000.

Example 2

We assume that in your FileMaker file the following fields are defined:

lastAccessedDate	date
gFilePath	Global, text

gFilePath should contain the path to the file, for example "D:\Logs\L2000_01.TXT" (Windows) or "Mac HD:Logs:Log 2000_01" (Mac). In a script add the following script step:

```
Set Field[lastAccessedDate, TrFile_GetDateLastAccessed( "-Unused" ; gFilePath )]
```

This will store the date the file, specified by gFilePath, was opend for the last time of the file in the field lastAccessedDate.

TrFile_GetDateModified

Syntax TrFile_GetDateModified(switches ; filePath)

Returns the modification date for the file specified by the filePath. The modification date is also displayed in the Finder or Explorer.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want the information

Returned result

The returned result is the modification date of the file. The date is in the same format as a date field.

An error code might also be returned. An error always starts with 2 dollars, followed by the error code. Returned error codes can be:

$$-43	fnfErr	File not found, check if the path is valid
$$-1	genericErr	The file could not be found (older versions of the plug-in)

Other errors may be returned.

Special considerations

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

Set Field[modificationDate, TrFile_GetDateModified("-Unused" ; "C:\Test.txt")]

This may return the number 730173 which is equivalent to the date: 23 Feb. 2000.

Example 2

We assume that in your FileMaker file the following fields are defined:

modificationDate	date
gFilePath	Global, text

gFilePath should contain the path to the file, for example "D:\Logs\L2000_01.TXT" (Windows) or "Mac HD:Logs:Log 2000_01" (Mac). In a script add the following script step:

```
Set Field[modificationDate, TrFile_GetDateModified( "-Unused" ; gFilePath )]
```

This will store the modification date of the file specified by gFilePath in the field modificationDate.

TrFile_GetFileAttribute

Syntax TrFile_GetFileAttribute(switches ; filePath)

Returns an attribute of the file specified by the filePath.

Parameters

switches	this determines the attribute that is returned
filePath	the path to the file

switches can be one of the following:

-LockAttr	return number indicating whether the file is locked (Mac) or Read-only (Windows)
-HiddenAttr	return number indicating whether the file is hidden (invisible)
-ArchiveAttr	return number indicating whether the file's archive bit is set (Windows only)
-LabelNumAttr	return number indicating the label of a file (Mac OS only)
-FindercommentAttr	return the finder comment of a file (Mac OS only)
-PackageFolderAttr	return number indicating whether the file is a package folder (Mac OS only)

Returned result

The returned result is the requested attribute of the file.

Result for switch "-LockAttr":
Returns 1 for a locked file and 0 for an unlocked file. On Windows it returns the equivalent "Read-only"-attribute: 1 if the file is a Read-only file and 0 otherwise.

Result for switch "-HiddenAttr":
Returns 1 if the file is hidden and 0 otherwise.

Result for switch "-ArchiveAttr":
Returns 1 if the file's archive bit is set and 0 otherwise. This bit is only available on Windows. On Mac OS this switch always returns 0.

Result for switch "-LabelNumAttr":
Returns a number between 0 (=no label) and 7. This attribute is only available on Mac OS. On Windows this switch always returns 0.

Result for switch "-FindercommentAttr":
Returns the finder comment, which is visible with the Get Info command in the Mac OS Finder. On Windows this switch always returns "".

Result for switch "-PackageFolderAttr":
Returns 1 if the item is a package folder and 0 otherwise. For example all Mac OS X applications are package folders. This attribute is only available on Mac OS. On Windows this switch always returns 0.

Note that also an error code might be returned. An error always starts with 2 dollars, followed by the error code. Returned error codes can be:

$$-43	fnfErr	File not found, check if the path is valid
$$-1	genericErr	The file could not be found (older versions of the plug-in)

Other errors may be returned.

Special considerations

The new switch "-PackageFolderAttr" can be used to determine if a result of the drag and drop action is a package.
NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

```
Set Field[result, TrFile_GetFileAttribute( "-LockAttr" ; "C:\test.doc" )]
```

TrFile_GetFileAttribute

The result will be 1 if the file is Read-Only and 0 otherwise.

Example 2

We assume that in your FileMaker file the following fields are defined:

locked	text
fileSpec	text

"fileSpec" should contain the path to an existing file, for example "D:\Out.txt" (Windows) or "Mac HD:Out.txt" (Mac). In ScriptMaker add the following script step:

```
Set Field[locked, TrFile_GetFileAttribute( "-LockAttr" ; fileSpec)]
```

This will get the locked status of the file into the field "locked".

TrFile_GetFileCreator

Syntax TrFile_GetFileCreator(switches ; filePath)

Returns the Creator for the file specified by the filePath.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want the information

Returned result

Creator: The Creator of a file is a 4 character code used to designate the application that created a file. For example: FileMaker Pro 7 creates database files with a Creator "FMP7".

Special considerations

•This function is not available in Windows.•
See also the function "TrFile-GetFileType" to get the FileType.
If the creator is empty the function returns: "0000" (4 zero's) .

Example usage

Set Field[result, TrFile_GetFileCreator("-Unused" ; "KES:readme")]

This will (probably) return "ttxt" which is the Creator for the SimpleText application.

Example 2

We assume that in your FileMaker file the following fields are defined:

creator	text
gFilePath	Global, text

gFilePath should contain the path to the file, for example "Mac HD:Logs:Log 1" (Mac). In a script add the following script step:

Set Field[creator, TrFile_GetFileCreator("-Unused" ; gFilePath)]

This will store the Creator of the file specified by gFilePath in the field "creator".

TrFile_GetFileHash

Syntax TrFile_GetFileHash(switches ; filePath)

Returns the MD5 or SHA1 hash value (checksum) of a file.

Parameters

switches	determines the behavior of the function
filePath	the path to the file of which you want the hash value

Switches must be one of this:

-MD5	use the MD5 algorithm to calculate the hash.
-SHA1	use the SHA-1 algorithm to calculate the hash.

You can also add this switch below:

-Colons	add colons between the result, for easy human reading

Returned result

the hash value, which is a text of 32 or 40 characters.

If the file does not exist or if an error occurs an error code will be returned, starting with $$:

$$-43	fnfErr	File or folder not found
$$-50	paramErr	Parameter error

Other error codes may be returned.

Special considerations

The function result is also called a message-digest. It provides an easy way to check if a file has not changed (by comparing it to a stored hash). Also you can use this hash to check if two files are identical.

Note: On Mac OS X only the data fork of a file is considered. The resource fork of a file is ignored.

SHA-1 is considered more secure as it is a longer hash (160-bit).

Example usage

TrFile_GetFileHash("-MD5" ; "C:\Test.txt")]

This will return the hash value which looks like this: "6858c159c8ed6c5957fdf344e3367495"

Example 2

Set Variable [$hash ; TrFile_GetFileHash("-MD5 -Colons" ; "filemac:/Mac HD/Users/Smith/report.txt")]

This will set the variable $hash to the MD5 hash value of the file. The result will be formatted, for easier reading, with colons between each byte and looks similar to this:

"65:da:d9:97:46:2d:21:95:a7:4e:e8:a1:fd:5f:3c:ed"

TrFile_GetFileSize

Syntax TrFile_GetFileSize(switches ; filePath)

Returns the file size for the file specified by the filePath. The size indicates the size the file is using on the disk, not the actual size of the data and resources.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want the information

Returned result

The size (in number of bytes) of the file on disk.

Special considerations

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

Assume there is a file "Read me" with the text "Hello!" in it. Then:

```
Set Field[MyTextField, TrFile_GetFileSize("-Unused" ;  "Disk:Read me")]
```

returns 1024 (= 1 Kb) which is the size used on disk.

Example 2

We assume that in your FileMaker file the following fields are defined:

gFileSpec	Global, text
gSize	Global, number

gFileSpec should contain the path to an existing file, for example "D:\Readme.txt" (Windows) or "Mac HD:Readme.txt" (Mac). In a script add the following scriptstep:

```
Set Field[gSize, TrFile_GetFileSize("-Unused" ; gFileSpec)]
```

This will fill gSize with the total size of the file.

TrFile_GetFileType

Syntax TrFile_GetFileType(switches ; filePath)

Returns the FileType for the file specified by the filePath.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want the information

Returned result

The FileType is a 4 character code used to designate the type of file. For example: FileMaker 7 files have a FileType of "FMP7" and Applications have FileType of "APPL".

Special considerations

This function is not available in Windows. Windows uses the 3 letter extension, like .fp7 instead.

If the file type is empty the function returns: "0000" (4 zero's) .

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

Set Field[result, TrFile_GetFileType("-Unused" ; "KES:test.fp7")]

This will (probably) return "FMP7" which is the FileType for FileMaker Pro 7.

Example 2

We assume that in your FileMaker file the following fields are defined:

fileType	text
gFilePath	Global, text

gFilePath should contain the path to the file, for example "Mac HD:Logs:Log 1" (Mac). In a script add the following script step:

```
Set Field[fileType, TrFile_GetFileType( "-Unused" ; gFilePath )]
```

This will store the FileType of the file specified by gFilePath in the field "fileType".

TrFile_GetIcon

Syntax TrFile_GetIcon(switches ; FileSpec ; {iconName})

Gets the icon of a file and returns it as a PNG image.

Parameters

switches	these determine the properties of the image that is returned
FilePath	the path to the file from which to get the icon
iconName	(optional) the (internal) name of the created PNG image

Switches can be empty or one or more of the following:

-Size=64	(default) maximum size of the height or width is 64 pixels
-Size=128	maximum size of the height or width is 128 pixels
-Size=256	maximum size of the height or width is 256 pixels

You can also add these switches:
-RetinaResolution create icons with double resolution (144 dpi), ideal for displaying on retina screens.

Returned result

The returned result is a PNG thumbnail of the icon. If an error occurs an error code is returned. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

$$-2003	cantFindHandler	No icon could be created

Other errors may be returned.

Special considerations

The optional iconName is added to the created image data and also when the image is stored in the container. This name is normally not visible, but will be used for example when you perform an Export Field Contents action later.

You can also use FileMaker styled paths, like "filemac:/MacHD/folder/myData.doc".

Example usage

```
Set Field [ container field, TrFile_GetIcon( "-Unused" ; "Mac HD:sample.pdf" )]
```

This will create a PNG thumbnail image of the icon of the PDF file and put it in a container field.

TrFile_GetPathTo

Syntax TrFile_GetPathTo(switches ; desiredFilename ; fileSize)

Returns the path to an object, at the moment the path to the FileMaker application or the current FileMaker file.

Parameters

switches	this determines which path is returned
desiredFilename	the name of the file to find
fileSize	the size of the file to find

switch can be only the following:

-CurrentApplication	return the FileSpec of the FileMaker application
-CurrentFileName	return the FileSpec of the FileMaker file that is currently active (the front window)
-CurrentAppFolder	return the folder of the current running (FileMaker) application.

Returned result

FileSpec: The function returns a FileSpec for the current database file.

The function can also return an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

$$-43	fnfErr	File not found.
$$-50	paramErr	There was an error with the parameter.
$$-4215	invalidFMPVersion	This function does not work with this version

Other errors may be returned.

Special considerations

The -CurrentApplication switch of this function works on all platforms and also with a runtime.
The -CurrentFileName switch of this function works on Windows from FileMaker version 5. On Mac OS it works from FileMaker 4.0 and later. If you use it on Windows under a version earlier than FileMaker 5 an error code "$$ -4215" (kErrInvalidFMPVersion) is returned.

IMPORTANT: When creating a runtime with FileMaker Developer on Windows the -CurrentFileName function does NOT work on Windows NT, Windows 2000 and Windows XP. It works with runtimes on Mac OS and Windows 95/98/Me.

Also when a file is hosted as a guest, for example with FileMaker Server it can not return the path, in this case it will return an error code "$$-43" (fnfErr = file not found).

NOTE FileMaker Pro 5.5 has a Status(CurrentFilePath) function, that also returns a path to the file. For FileMaker 7 and later this function is called: Get(FilePath)

The switch -CurrentAppFolder returns the folder of the current running application. This is for example the folder of the FileMaker Pro application or the folder of the FileMaker runtime application.

IMPORTANT (Mac OS X only) Version 2.7 changed the behaviour of the switch "-CurrentApplication". This function now returns the FSSpec to the FileMaker application package (usually FileMaker Pro.app), instead of the carbon executable inside this package.
So the full path will no longer be:

"MacHD:Applications:FileMaker Pro 6 Folder:FileMaker Pro.app:Contents:MacOS:FileMaker Pro"

Instead it will be:

"MacHD:Applications:FileMaker Pro 6 Folder:FileMaker Pro.app:"

Be aware that the FSSpec of this package is a folder! This may break a script if you assumed the "Contents:MacOS: FileMaker Pro" was at the end. If you only want to find the folder of the FileMaker application use the new switch "-CurrentAppFolder".

TrFile_GetPathTo

"-CurrentAppFolder" will return the path with a backslash (Windows).

Example usage

Set Field [result, TrFile_GetPathTo("-CurrentApplication")]

This will get the path to the current FileMaker application. An example result can be on Mac "MyDisk:Applications (Mac OS 9):FileMaker Pro 5.5 Folder:FileMaker Pro". On Windows this might be: "C:\Program Files\FileMaker\FileMaker Pro 5.5 Folder\FileMaker Pro.exe"

Example 2

```
Set Field [result, TrFile_GetPathTo(
        "-CurrentFileName" ; Get(FileName) ; Get(FileSize))]
```

This will get the path to the current FileMaker file that is open. An example result can be on Mac "MyDisk:Documents:test.fp7". On Windows this might be: "D:\Documents\test.fp7"

TrFile_GetResForkSize

Syntax TrFile_GetResForkSize(switches ; filePath)

Returns the size of the resource fork for the file specified by the filePath. The size indicates the actual number of bytes used by the resource fork.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want the information

Returned result

The size (in bytes) of the resource fork of the file specified. On Mac a file can have a data fork and also a resource fork. In this fork resources, like for example pictures are stored.

Special considerations

This function is not available in Windows.

Example usage

Assume there is a file "Read me.txt" with the text "Hello!" in it. Then:

```
Set Field[MyTextField, TrFile_GetResForkSize( "-Unused" ; "Disk:Read me.txt")]
```

might return 453 which is the size of the resource fork.

Example 2

We assume that in your FileMaker file the following fields are defined:

gFileSpec	Global, text
gSize	Global, number

gFileSpec should contain the path to an existing file, for example "Mac HD:Readme.txt". In a script add the following scriptstep:

```
Set Field[gSize, TrFile_GetResForkSize( "-Unused" ;  gFileSpec)]
```

This will fill gSize with the resource fork size of the file.

TrFile_GetTimeCreated

Syntax TrFile_GetTimeCreated(switches ; filePath)

Returns the creation time for the file specified by the filePath.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want the information

Returned result

The returned result is the creation time of the file. The time is in the same format as a time field (number of seconds since 00:00:00).

An error code might also be returned. An error always starts with 2 dollars, followed by the error code. Returned error codes can be:

$$-43	fnfErr	File not found, check if the path is valid
$$-1	genericErr	The file could not be found (older versions of the plug-in)

Other errors may be returned.

Special considerations

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

Set Field[creationTime, TrFile_GetTimeCreated("-Unused" ; "C:\Test.txt")]

This might return the number 7264 which is equivalent to the time: 02:01:04.

Example 2

We assume that in your FileMaker file the following fields are defined:

creationTime	time
gFilePath	Global, text

gFilePath should contain the path to the file for example "D:\Logs\L01.TXT" (Windows) or "Mac HD:Logs:Log 1" (Mac). In a script add the following script step:

```
Set Field[creationTime, TrFile_GetTimeCreated( "-Unused" ;  gFilePath )]
```

This will store the creation time of the file specified by gFilePath in the field creationTime.

TrFile_GetTimeLastAccessed

Syntax TrFile_GetTimeLastAccessed(switches ; filePath)

Returns the time a file was accessed (opened) for the last time.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want the information

Returned result

The returned result is the time the file was last accessed. The time is in the same format as a time field (number of seconds since 00:00:00).

An error code might also be returned. An error always starts with 2 dollars, followed by the error code. Returned error codes can be:

$$-43	fnfErr	File not found, check if the path is valid
$$-1	genericErr	The file could not be found (older versions of the plug-in)

Other errors may be returned.

Special considerations

On Windows some file systems (FAT) only store the actual Date of the last access. The time in this case will be 00:00:00. See the TrFile_GetDateLastAccessed in this case.

Example usage

Set Field[lastAccessedTime, TrFile_GetTimeLastAccessed("-Unused" ; "C:\Test.txt")]

This might return the number 7264 which is equivalent to the time: 02:01:04.

Example 2

We assume that in your FileMaker file the following fields are defined:

lastAccessedTime	time
gFilePath	Global, text

gFilePath should contain the path to the file for example "D:\Logs\L01.TXT" (Windows) or "Mac HD:Logs:Log 1" (Mac). In a script add the following script step:

```
Set Field[lastAccessedTime, TrFile_GetTimeLastAccessed( "-Unused" ; gFilePath )]
```

This will store the time the file, specified by gFilePath, was openend for the last time of the file in the field lastAccessedTime.

TrFile_GetTimeModified

Syntax TrFile_GetTimeModified(switches ; filePath)

Returns the modification time for the file specified by the filePath. The modification time is also displayed in the Finder or Explorer.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want the information

Returned result

The returned result is the modification time of the file. The time is in the same format as a time field (number of seconds since 00:00:00).

An error code may also be returned. An error always starts with 2 dollars, followed by the error code. Returned error codes can be:

$$-43	fnfErr	File not found, check if the path is valid
$$-1	genericErr	The file could not be found (older versions of the plug-in)

Other errors may be returned.

Special considerations

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

Set Field[modificationTime, TrFile_GetTimeModified("C:\Test.txt")]

This might return the number 7265 which is equivalent to the time: 02:01:05.

Example 2

We assume that in your FileMaker file the following fields are defined:

modificationTime	time
gFilePath	Global, text

gFilePath should contain the path to the file, for example "D:\Logs\L01.TXT" (Windows) or "Mac HD:Logs:Log 1" (Mac). In a script add the following script step:

```
Set Field[modificationTime, TrFile_GetTimeModified( gFilePath )]
```

This will store the modification time of the file specified by gFilePath in the field modificationTime.

TrFile_GetTimestampCreated

Syntax TrFile_GetTimestampCreated(switches ; filePath)

Returns the creation date and time for the file specified by the filePath.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want the information

Returned result

The returned result is the creation date and time of the file. The result is formatted like this:
YYYY-MM-DD HH:MM:SS. Store the result in a text field.

An error code might also be returned. An error always starts with 2 dollars, followed by the error code. Returned error codes can be:

$$-43	fnfErr	File not found, check if the path is valid
$$-1	genericErr	The file could not be found (older versions of the plug-in)

Other errors may be returned.

Special considerations

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

Set Field[result, TrFile_GetTimestampCreated("-Unused" ; "C:\Test.txt")]

This may return the result: "2000-02-22 15:55:17".

Example 2

We assume that in your FileMaker file the following fields are defined:

creationDateTime	text
gFilePath	Global, text

gFilePath should contain the path to the file, for example "D:\Logs\L01.TXT" (Windows) or "Mac HD:Logs:Log File 1" (Mac). In a script add the following script step:

```
Set Field[creationDateTime, TrFile_GetTimestampCreated( "-Unused" ;  gFilePath )]
```

This will store the creation date and time of the file specified by gFilePath in the field creationDateTime.

TrFile_GetTimestampLastAccessed

Syntax TrFile_GetTimestampLastAccessed(switches ; filePath)

Returns the date and time a file was accessed (opened) for the last time.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want the information

Returned result

The returned result is the date and time the file was last accessed. The result is formatted like this:
YYYY-MM-DD HH:MM:SS. Store the result in a timestamp field.

An error code might also be returned. An error always starts with 2 dollars, followed by the error code. Returned error codes can be:

$$-43	fnfErr	File not found, check if the path is valid

Other errors may be returned.

Special considerations

On Windows some file systems (FAT) only store the actual date of the last access. The time in this case the time part will be 00:00:00..

Example usage

Set Field[lastAccessedTimestamp, TrFile_GetTimestampLastAccessed("-Unused" ; "C:\Test.txt")]

This may return the result: "2006-12-22 15:55:17".

Example 2

We assume that in your FileMaker file the following fields are defined:

lastAccessedDateTime	timestamp
gFilePath	Global, text

gFilePath should contain the path to the file for example "D:\Logs\L01.TXT" (Windows) or "Mac HD:Logs:Log 1" (Mac). In a script add the following script step:

Set Field[lastAccessedDateTime, TrFile_GetTimestampLastAccessed("-Unused" ; gFilePath)]

This will store the timestamp the file, specified by gFilePath, was opend for the last time of the file in the field lastAccessedTime.

TrFile_GetTimestampModified

Syntax TrFile_GetTimestampModified(switches ; filePath)

Returns the modification date and time for the file specified by the filePath. The modification date and time are also displayed in the Finder or Explorer.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file for which you want the information

Returned result

The returned result is the modification date and time of the file. The result is formatted like this:
YYYY-MM-DD HH:MM:SS. Store the result in a text field.

An error code might also be returned. An error always starts with 2 dollars, followed by the error code. Returned error codes can be:

$$-43	fnfErr	File not found, check if the path is valid
$$-1	genericErr	The file could not be found (older versions of the plug-in)

Other errors may be returned.

Special considerations

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

Set Field[result, TrFile_GetTimestampModified("-Unused" ; "C:\Test.txt")]

This may return the result: "2000-12-31 15:55:17".

Example 2

We assume that in your FileMaker file the following fields are defined:

modificationDateTime	text
gFilePath	Global, text

gFilePath should contain the path to the file, for example "D:\Logs\L2000_01.TXT" (Windows) or "Mac HD:Logs:Log 2000_01" (Mac). In a script add the following script step:

Set Field[modificationDateTime, TrFile_GetTimestampModified("-Unused" ; gFilePath)]

This will store the modification date and time of the file specified by gFilePath in the field modificationDateTime.

TrFile_GetTypeOfItem

Syntax TrFile_GetTypeOfItem(switches ; filePath)

Returns the type of an item, for example if it is a folder or a file.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file or folder for which you want to know the type

Returned result

The returned result can be one of:

file	if the path points to a file
folder	if the path points to a folder (directory)
package folder	if the path points to a package folder (Mac OS X only)

If the path does not exist or if an error occurs an error code will be returned, starting with $$:

$$-43	fnfErr	File or folder not found
$$-50	paramErr	Parameter error

Other error codes may be returned.

Special considerations

This function provides an easy way to see if a path points to a folder or a file. Please note that in the future other types may be returned.

About package folders: these are folders that behave more like a file. For example all Mac OS X applications are package folders.

Example usage

TrFile_GetTypeOfItem("-Unused" ; "C:\Test.txt")]

This will return "file" if the file exists (and is a file). If it does not exists $$-43 is returned.

Example 2

Set Variable [$whatKind ; TrFile_GetTypeOfItem("-Unused" ; "filemac:/Mac HD/Users/Smith/Data Folder)]

This will set the variable $whatKind to "folder" if the folder exists (and is a folder). If it does not exists $$-43 is returned.

TrFile_GetZipInfo

Syntax TrFile_GetZipInfo(switches ; sourceZipFilePath)

Get information from a ZIP file.

Parameters

switches	modifies the behavior of the function
sourceZipFilePath	the path to the ZIP file

The switches parameter currently has one possible value:

-ItemList	(default) return the list of files and folders in the ZIP file

You can also add this switch:

-SkipHiddenMacOSXItems	don't list items which are in a __MACOSX folder in the ZIP Mac OS X stores extra info and resource forks in these __MACOSX folders

Returned result

item list a list of files and folders in the ZIP file.

The result can also be an error code. Possible error codes are:

$$-43	fnfErr	ZIP file not found

Other error codes can be returned.

Special considerations

Each item is returned on a separate line: all items (including the last item) are followed by a return character.

Example usage

```
Set Field[ $ZIP_Items ; TrFile_GetZipInfo( "-ItemList" ; "D:\myZIP2016.zip") ]
```

This will list what is stored in the ZIP file. The result will be for example:

```
my report/
my report/fiscal.xls
my report/illustration.jpg
my report/extra data/
my report/extra data/note.txt
```

In this list you see that the folder "my report" contains 2 files and a subfolder "extra data". Folders end with a slash and subfolders are also separated with a slash.

Example 2

We assume that in your FileMaker file the following fields are defined and filled with a path to the source ZIP file:

this::ZIPsourcePath	may contain something like "MacHD:Users:kip:bigtext.zip"
this::ZIPlist	the result will be stored here

This example will first set the variable from the path stored in the first field and then list the items of the ZIP file.
In a script add the following steps:

```
#Initialize: copy the field into a variable first:
Set Variable [ $SourcePath ; this::ZIPsourcePath ]
#Now get the item list :
Set Variable [ $ItemList ; TrFile_GetZipInfo( "-ItemList" ; $SourcePath ) ]
If[ Left ($ItemList ; 2) <> "$$"
    Set Field[this::ZIPlist ; $ItemList]
```

```
Else
        Show Custom Dialog[ "An error occurred" ; "OK" ]
Endif
```

TrFile_InsertContents

Syntax TrFile_InsertContents(switches ; sourceFilePath; destFilePath ; insertText ;{ position })

Inserts text into a file, at the specified position.

Parameters

switches	these alter the behaviour of the function
sourceFilePath	the path to the source file
destFilePath	(optional) the path to the destination file
insertText	text to insert into the file
position	(optional) position where to insert the text. This is calculated in number of bytes from the beginning.

Switches can be empty or one of this:

-Encoding=Native	(default) native to the platform
-Encoding=UTF8	
-Encoding=ASCII_DOS	
-Encoding=ASCII_Windows	
-Encoding=ASCII_Mac	
-Encoding=ISO_8859_1	(Windows Latin-1)
-Encoding=BytesOnly	only characters in the range 0-255 are inserted, others are skipped

This determines the character encoding of the inserted data.

You can also add this switch:

-CRtoCRLF Use Windows line endings. Returns in the text parameter are substituted with CRLF (Carriage Return followed by a Line Feed) in the written file.

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The string of characters was written in the file
$$-43	fnfErr	Source file not found

Other errors may be returned.

Special considerations

If the position parameter is 0 or is empty, the text will be inserted at the beginning of the file.

The default is -Encoding=Native, which is the current encoding as set in the system preference, for example on western Windows systems it usually is ISO_8859_1 or ASCII_Windows. On Mac OS X the default will be ASCII_Mac.
If the source file is a different encoding it is best to use that encoding.
For most situations it is best to insert with encoding UTF-8.

If the destination parameter is empty, the insertion text will be saved in the source file itself. This happens also when you specify the same source file and destination file.

Example usage

Set Field[MyTextField, TrFile_InsertContents("-Unused" ; "HD Mac:data:My Letter"; "HD Mac:data::My newLetter"; "my extra text"; 42)]

This will insert the text "my extra text" after the 42th byte and save the result into the new file "HD Mac:data::My newLetter".

TrFile_Launch

Syntax TrFile_Launch(switches ; filePath ; { params })

Opens a file with the application that has registered it. This is the same application that would open it if you would manually double-click it.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file to be launched
params	(optional) extra parameters to be sent to the launching application (Windows only)

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The file was opened with its application
$$-35	nsVolErr	No such volume (Wrong disk name or not mounted).
$$-43	fnfErr	File not found.
$$-50	paramErr	Parameter error.
$$-1	genericErr	The file could not be opened.

Other errors may be returned.

Special considerations

On the Mac the program that opens the file is determined by the FileType of the file.
On Windows this is determined by the 3 letter extension of the file.
So a text file "ReadMe.txt" will usually be opened by SimpleText (Mac) or WordPad (Windows).

On Windows you can add extra parameters to the launch command, which are then sent to the launching application.

Example usage

Set Field[result, TrFile_Launch("-Unused" ; "C:\readme.doc")]

This will open the file in the application Microsoft Word.

Example 2

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gFilePath	Global, text

gFilePath should contain the path to the file, for example "D:\Logs\L01.TXT" (Windows) or "Mac HD:Logs:Log 1" (Mac).
In a script add the following script step:

```
Set Field[gErrorCode, TrFile_Launch( "-Unused" ; gFilePath )]
```

This will launch the file specified by gFilePath with its application.

TrFile_ListDisks

Syntax TrFile_ListDisks(switches)

Lists the names of all currently available disks.

Parameters

switches (optional) determine way the result is returned

Switches can be empty or:
-ReturnAtEnd add an extra return character after the last found item in the list

Returned result

disknames a list of names of all the disks that are currently mounted (available).

Special considerations

Use FindFolder to find the startup disk.

Example usage

Set Field[result, TrFile_ListDisks("")]

On Mac OS this will return a list of names of all the disks that are currently mounted, for example:

KES¶
SPOCK¶
Photos till 2004¶
Fotos tm 991115

On Windows the drive letters of the available disks are returned, for example:

A:¶
C:¶
D:¶
K:

TrFile_ListFolder

Syntax TrFile_ListFolder(switches ; folderPath)

Lists what is inside a folder (directory). It will return all files and/or folders, depending on the switches given.

Parameters

switches	these determine the items that are listed
folderPath	the path to the folder to list

switches can be one or more of the following:

-Files	list all files in this folder
-Folders	list all folders (subdirectories) in this folder
-Showaliases	list all aliases (shortcuts) in this folder
-Showshortcuts	list all aliases (shortcuts) in this folder (you can use the one you like)
-Showinvisibles	list all invisible files and folders
-Showpointdirs	Windows: list also the directory . (current dir) and .. (parent dir)
-Showpointdirs	Mac OS: this switch is ignored
-Longnames	(default) Mac OS X: return long names, possibly more than 31 characters
-Shortnames	Mac OS X: return shortened names, less than 31 characters long
-Recursive	get the list of files and folders and all the subfolders of the folder you are listing
-ShowPackageContents	(Mac OS X) also list the contents of a package folder (used in combination with -Recursive)
-DontEncodeSlash	(Mac OS X) dont encode slashes in name with %2F (in combination with -Recursive)
-ReturnAtEnd	add an extra return character after the last found item in the list

Returned result

folder list a list of names of the items separated by returns.

Special considerations

You can recursively get the list of files and folders and all the subfolders of the folder you are listing. To enable this add the switch "-Recursive". Note that files and folders in subfolders will be listed with the relative folderpath, with a slash as separator, for example "SubFolder1/filename.txt".
Starting with v5.5 the switch "-Longnames" is the default, so you no longer need to specify it. If you want short names (on Mac OS X): use the "-Shortnames" switch. Mac OS 9.x supported only names up to 31 characters, so with the "-Shortnames" switch the names will be shorter or equal than 31 characters, for longer names the file name will look like mygreatimagefilebuttoo#E11E.txt

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

Set Field [result, TrFile_ListFolder("-Files -Folders " ; "Mac HD:")]

This will return a list of all files and folders on the hard disk "Mac HD". Note that the order of the switches is not relevant. This might be returned:

Test.fp7¶
Desktop Folder¶
Program files¶
etc...

On Windows an example usage is:
Set Field [result, TrFile_ListFolder("-Files -Showshortcuts" ; "C:\Data\")]

TrFile_ListFolder

This will return a list of all files and shortcuts on the hard disk "C:\Data\".

Example 2

We assume that in your FileMaker file the following fields are defined:

gFolderPath	Global, text
gFolderList	Global, text

gFolderPath should contain the path to the folder, for example "D:\Logs\Data\" (Windows) or "Mac HD:Logs:Data:" (Mac OS). In a script add the following scriptstep:

```
Set Field[gFolderList, TrFile_ListFolder( "-Files -Folders " ; gFolderPath )]
```

This will list the contents of the folder specified in the gFolderPath.

TrFile_MetaData

Syntax TrFile_MetaData(switches ; filePath)

Gets metadata out of a graphic file. Metadata can be inserted by for example Photoshop, and can contain a description of the image, copyright information, etc.

Parameters

switches these determine the metadata that is returned
filePath the path to the file

switches can be one or more of the following:
-GetIPTC get the IPTC metadata (Photoshop's File Info)
-GetXMP get XMP metadata (Adobe's Extensible Metadata Platform)
-GetExif get the Exif metadata (A standard for Digital camera's)
-GetRawExif get the Exif metadata (A standard for Digital camera's), in a more original form
-GetGPS get the GPS (Global Positioning System) metadata, with the image coordinates
-GetJPEGComment get the JPEG comment metadata of a JPEG file
-GetImageURL get the URL metadata (Photoshop's File Info extension)
-GetImageDescription get a description like width, height and depth, resolution, codec name etc.

-SourceMacCharSet the IPTC data has Mac encoded characters
-SourceWinCharSet the IPTC data has Windows encoded characters
-DontUseAutoUTF8Detection disable the automatic detection of UTF8 encoding

For movies these switches can be used:
-GetMovieDescription get a description like width, height and depth, timecode, frames per second, codec name etc.
-GetMovieDuration get the duration in seconds of a movie
-GetTimecodeBegin get the begin Timecode of a movie
-GetTimecodeEnd get the end Timecode of a movie
-GetTimecodeCurrent get the current Timecode of a movie

If there is no Timecode track an empty string is returned.

For PDFs this switch can be used:
-GetPDFDescription get metadata of a PDF file, such as the document’s title, author, and creation date

Returned result

metadata the requested metadata, if available.

If not available an error is returned:
$$-2026 userDataItemNotFound

Special considerations

Note that the data is returned in a raw form. See the example files "IPTC_XMP_Metadata" and GetEXIF" for information on how to parse this.

Exif (Exchangeable image file) metadata in an image file will contain annotation data such as the time a photo was created, aperture and shutter speed. Your digital camera needs to have put this in the file.

From v6.0: When using the switch "-GetIPTC" the plug-in can now automatically detect if the text is UTF-8 encoded (and also returns with the metadata text correctly formatted). If UTF-8 format is detected it will override the switches: "-SourceMacCharSet" and "-SourceWinCharSet".

IPTC data in image files can also be either Mac or Windows encoded . With the switches "-SourceMacCharSet" and "-SourceWinCharSet" you can manually tell the plug-in which character encoding you expect in the image files.

TrFile_MetaData

Alternatively you can retrieve both encodings and then see which is best. Note: If you don't specify a CharSet switch, the plug-in uses the encoding of the platform it is running on.

Version 5.5 added getting Exif from RAW files. v4.5 added the -GetRawExif switch: this will return the Exif data in a more original form, as fractions. For example aperture will be returned as 95/32, instead of 2.96875. This makes it possible to write Exif data back in the original form, with the TrFile_SetMetaData function.

Example usage

```
Set Field[result, TrFile_MetaData( "-GetExif " ; "C:\myData\Photo1.jpg" ) ]
```

This will return the raw EXIF data. The data will look like this:

```
##EXIF 270
OLYMPUS DIGITAL CAMERA
##END 270
##EXIF 306
2011:06:09 10:50:40
##END 306
...
```

Example 2

```
Set Field[result, TrFile_MetaData( "-GetIPTC " ;  gFilePath) ]
```

This will return the raw IPTC data. The data will look like this:

```
##IPTC 120
FOR IMMEDIATE RELEASE--FILE--Shania Twain performs at the Country Music Association Awards show in
Nashville, Tenn., ...
##END 120
##IPTC 122
CJC RWP MAH
##END 122
##IPTC 105
SHANIA TWAIN
##END 105
##IPTC 40
FOR IMMEDIATE RELEASE. A SEPT. 22, 1999 FILE PHOTO. DIGITAL IMAGE
##END 40
...
```

TrFile_MountDisk

Syntax TrFile_MountDisk(switches ; zone ; server; disk; { username ; password })

Connect to a shared network volume, by mounting the disk. You can then for example read or save files on this disk.

Parameters

switches	this modifies the behavior of the function
zone	(optional) the name of the AppleTalk zone (on Windows this parameter is not used)
server	on Mac the name of the Server, on Windows the UNC name of the share
disk	on Mac the name of the disk, on Windows (optional) the drive letter to use
username	(optional) the name of the user (if empty guest access is tried)
password	(optional) the password of this user

Switches can be empty or one of this on Mac OS X:

-TwoWayEncryptPassword	(default) use a 2 way encrypted password to logon
-OneWayEncryptPassword	use a 1 way encrypted password to logon
-PlainPassword	send the password in plain text

Switches can be empty or one of this on Windows:

-ReUseDriveLetter	(default) if the network disk already is mapped reuse that drive letter
-AlwaysNewDriveLetter	always map a new drive letter

Returned result

On Mac OS The returned result is an error code. If no error occurs a zero is returned.
On Windows the returned result is the drive letter or an error code. An example drive letter can be: "F:"

An error always starts with 2 dollars, followed by the error code. You should always check for errors.

$$-35	no such volume, disk not found
$$-50	parameter error, check if the supplied parameters are correct
$$-5023	the user is not authorized for this volume
$$-5000	Insufficient access privileges for operation
$$67	(Windows) the specified UNC name is invalid or can not be found.
$$1200	(Windows) the specified drive letter is invalid. enter a valid drive letter, like "K:".
$$2250	(Windows) not connected to a network

Other errors can be returned.

Special considerations

Due to differences in operating systems this function behaves a bit different on Mac compared to Windows. On Windows there is no zone name for example. See the platform specific examples below.

On Mac OS the plug-in also supports URL style AFP (AppleTalk File Protocol) servers. See also the examples below.

Regarding sending passwords on Mac OS: The plug-in always tries to use 2 way encrypted passwords. If however the server does not support this you can specify to use a one way encrypted password or a plain password, by giving the -OneWayEncryptPassword or -PlainPassword switch respectively.

On Windows the "server" parameter should be a UNC name, formatted like "\\servername\share". Also on Windows the parameter "disk" indicates the drive letter to use, like "K:". If empty the function picks a free drive letter.

IMPORTANT On some Windows implementations the username and password parameter have been switched. If you get a authorization error, try reversing the username and password!

After mounting the shared disk icon should now appear on your desktop (on Mac OS) or in the My Computer folder (on Windows). Depending on your access privileges, you can copy files, create folders and use items on the shared disk as if they were on your local hard disk.

TrFile_MountDisk

Example usage

Mac OS X:
Set Field [result, TrFile_MountDisk("-Unused " ; "MacZone" ; "McServer" ; "Shared Disk" ; "George" ; "secret")
Set Field [result,TrFile_MountDisk("-Unused " ; "" ; "McServer" ; "Guest folder")

The first set field step will mount the hard disk "Shared Disk" of server "McServer" in zone "MacZone" using the user name "George" with password "secret".
The second set field step will mount the volume "Guest folder" of server "McServer" using guest access. As the zone parameter is empty, the server can be in any zone.

The plug-in also supports URL style AFP servers names. For example a server on the intranet:
Set Field [result, TrFile_MountDisk("-Unused" ; "" ; "afp://192.168.1.55" ; "Shared Disk")]

Starting with version 2.6.1 the plug-in also supports URL style WebDAV servers names on Mac OS X 10.2 and later. For example an iDisk server:
Set Field [result, TrFile_MountDisk(
"-Unused" ; "" ; "http://idisk.mac.com/youridiskname" ; "")]

You can also connect to servers on the public Internet, for example:
Set Field [result, TrFile_MountDisk("" ; "" ; "afp://tramp.tr.fh-hannover.de" ; "Oeffentliche_Software")]

On Windows:

Set Field [result,
TrFile_MountDisk("-ReUseDriveLetter" ; "unused zone" ; "\\Troi Server\Projects" ; "" ; "George" ; "secret")]
Set Field [result,
TrFile_MountDisk("-Unused " ; "unused zone" ; "\\Troi Server\Projects" ; "Z:")]

The first set field step will mount the share "Projects" of server "Troi Server". using the user name "George" with password "secret". The second set field step will mount the same share using guest access, and mounting it as drive "Z:".

TrFile_MoveFile

Syntax TrFile_MoveFile(switches ; sourceFilePath ; destinationFilePath)

Moves (or renames) a file from one disk location to another.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
sourceFilePath	the path to the file to move
destinationFilePath	the path where the file must be moved to

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The file was moved
$$-43	fnfErr	Source file not found, or destination directory does not exist
$$-48	dupFNErr	Destination file already exists
$$-4254	returnInPathErr	The path contains an illegal return character
$$-1	genericErr	The file could not be moved

Other errors may be returned.

Special considerations

TIP: You can also use this function to rename a file
See also the functions "TrFile_SelectFileDialog and TrFile_SaveFileDialog" to get a FileSpec for a file.

This function only works when the source and destination file are on the same disk. If this is not the case, use the TrFile_CopyFile function and (optionally) use the TrFile-DeleteFile function to delete the source file.

Example usage

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gSourceFilePath	Global, text
gDestFilePath	Global, text

gSourceFilePath should contain the path to an existing file, for example "D:\Logs\Log01.TXT" (Windows) or "Mac HD:Logs:Log 1" (Mac). gDestFilePath should contain the path to the destination and should not exist for example "D:\New\L2000_01.TXT" (Windows) or "Mac HD:New:Log 2000_01" (Mac). In a script add the following scriptstep:

```
Set Field[gErrorCode, TrFile_MoveFile( "-Unused" ; gSourceFilePath ; gDestFilePath )]
```

This will move the source file to the path indicated in the gDestFilePath.

Example 2

Renaming a file: assume your C disk already contains a file "Testtext.txt". We assume that a global number field gErrorCode is defined. Create the following script:

```
Set Field[gErrorCode, TrFile_MoveFile( "-Unused" ; "C:\Testtext.txt" ; "C:\NewName.txt" )]
```

This script will move (rename) the file "Testtext.txt" to the "NewName.txt" file in the same directory. Note that on Mac OS the paths will be like: "Work Disk:NewName.txt".

TrFile_MoveFolder

Syntax TrFile_MoveFolder(switches ; sourceFolderPath ; destinationFolderPath)

Moves a folder to the specified folder path. Can also be used to rename the folder.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
sourceFolderPath	the path to the source folder to move
destinationFolderPath	the path where the folder must be moved to

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The folder was moved to the new place
$$-43	fnfErr	Source folder not found, or destination folder does not exist
$$-48	dupFNErr	Destination folder already exists
$$-122	badMovErr	The destination folder is inside the source folder
$$-1407	errFSNotAFolder	The source is not a folder

Other errors may be returned.

Special considerations

You can use the function TrFile_DeleteFolder to delete a folder first if it exists at the destination.

On Mac this function only works when the source and destination folder are on the same disk. If this is not the case, use the TrFile_CopyFolder function and (optionally) use the TrFile-DeleteFolder function to delete the source folder.

Example usage

Assume your C disk already contains a folder "TestFolder" and a folder "MyFiles". We assume that a global number field gErrorCode is defined. In ScriptMaker create the following script:

```
Set Field[gErrorCode, TrFile_MoveFolder( "-Unused" ; "C:\TestFolder" ; "C:\MyFiles\MovedFolder" )]
```

This script will move the folder"TestFolder" and its contents to the "MovedFolder" file inside the folder "MyFiles" on the C: disk. On Mac OS X the paths will be of the form "Mac HD:MyFiles:MovedFolder".
Notice that the destination folder has been renamed.

Example 2

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gSourceFolderPath	Global, text
gDestFolderPath	Global, text

gSourceFolderPath should contain the path to an existing folder, for example "D:\Logs\" (Windows) or "Mac HD:Logs:" (Mac). gDestFolderPath should contain the path to the destination and should not exist, for example "D:\Logs\L2007_12" (Windows) or "Mac HD:Logs:Log 2007_12" (Mac). In a script add the following script step:

```
Set Field[gErrorCode, TrFile_MoveFolder( "-Unused" ; gSourceFolderPath ;  gDestFolderPath )]
```

This will move the source folder to the path indicated in the gDestFolderPath.

TrFile_Reveal

Syntax TrFile_Reveal(switches ; filePath)

Reveals a file (or folder) in the Finder or Explorer.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file to be revealed

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The file was revealed in the Finder or Explorer
$$-35	nsVolErr	No such volume (Wrong disk name or not mounted)
$$-43	fnfErr	File not found
$$-50	paramErr	Parameter error

Other errors may be returned.

Special considerations

You can reveal files ànd folders!
NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

Set Field[result, TrFile_Reveal("-Unused" ; "C:\readme.doc")]

This will make the file visible in a Windows Explorer window and select it.

Example 2

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gFilePath	Global, text

gFilePath should contain the path to the file, for example "D:\Logs\L01.TXT" (Windows) or "Mac HD:Logs:Log 1" (Mac). In a script add the following script step:

```
Set Field[gErrorCode, TrFile_Reveal( "-Unused" ;  gFilePath )]
```

This will show the file specified by gFilePath in the Finder or Explorer. It will open the enclosing folder and show the files selected.

TrFile_SaveFileDialog

Syntax TrFile_SaveFileDialog(switches ; prompt ; { defaultName ; { initialFolder }})

Presents the user with a dialog, in which the user can specify where to save a file. The function returns with the FileSpec of the chosen file. Use this FileSpec for other File plug-in functions.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
prompt	the prompt that will be displayed to instruct the user
defaultName	(optional) the default file name of the file to be saved
initialFolder	(optional) the path to the folder where the dialog initially starts

Returned result

FileSpec: The function returns a FileSpec for the selected file.

Use this FileSpec as the basis for other File plug-in functions.

If the user cancels an error code of "$$-1" is returned.

Special considerations

New: filtering now also available on Windows! To change the default filtering see the TrFile_SetDefaultType function, or take a look at "Filtering Files" in the example databases.

Example usage

```
Set Field[myFileName, TrFile_SaveFileDialog(
                        "Please choose a file to create" ; "myFile.txt" ; gInititialDir )]
If [Left(myFileName, 2) <> "$$"]
  Set Field[gErrorCode, TrFile_CreateFile( myFileName )]
End If
```

This will ask the user for a file to create, which is then created.

TrFile_Search

Syntax TrFile_Search(switches ; folderPath ; searchName)

Search a folder (or volume) for a file or folder (directory).

Parameters

switches	these determine the items that are listed
folderPath	the path to the folder (or volume) in which to search
searchName	the (part of the) filename or foldername you want to find

switches can be one or more of the following:

-Files	search for files
-Folders	search for folders
-Exactname	the filename must exactly match the searchname
-Showaliases	search also aliases (shortcuts)
-Showshortcuts	search also aliases (shortcuts) (you can use the one you like)
-Showinvisibles	search also invisible files and folders
-Shortnames	return shortened names, less than 31 characters long (for classic Mac OS)
-ReturnAtEnd	add an extra return character after the last found item in the list
-Exhaustive	search by enumerating the folders, might be slow for folders with lots of items
-UseSpotlight	(OS X only) search for files and folders using Spotlight

Returned result

paths a list of paths of the found items separated by returns.

Special considerations

You can also specify a folder in which the search must be done, instead of searching the whole volume.
Note that the final path is also followed by a return!

You can now search using Spotlight on OS X by adding the switch -useSpotlight. But note that Spotlight does not return all files, for example it omits hidden (invisible) files. Also Spotlight search will not work if a disk is not indexed yet. It is however quite fast.

Starting with v5.5 the switch "-Longnames" is the default, so you no longer need to specify it. If you want short names (on Mac OS X): use the "-Shortnames" switch. Mac OS 9.x supported only names up to 31 characters, so with the "-Shortnames" switch the names will be shorter or equal than 31 characters, longer filenames will look like mygreatimagefilebuttoo#E11E.txt

Example usage

```
Set Field [ result, TrFile_Search( "-files -folders" ; "HD:Users:Kes:" ; "readme" )]
```

This will return a list of all files and folders in folder "HD:Users:Kes:" with the name 'readme'. Note that the order of the switches is not relevant, and also the & is just there for readability. This might be returned:

```
HD:Users:Kes:Foto, film, geluid:Geluiden:Alert!!!:Readme¶
HD:Users:Kes:Programma's:Beeld:GIFConverter 2.3.7 ƒ:README-registration¶
HD:Users:Kes:Programma's:Beeld:MOVIES:OBJTOOL:README, Make QTVR Object¶
etc...
```

On Windows an example use is:

```
Set Field [ result, TrFile_Search( "-files -folders" ; "C:" ; "readme" )]
```

Example 2

TrFile_Search

Set Variable [$result, TrFile_Search("-files -folders -invisibles" ; "HardDisk:" ; "readme")]

This will return a list of all files and folders on the hard disk "HardDisk" with the name 'readme', including invisible files.

TrFile_SelectFileDialog

Syntax TrFile_SelectFileDialog(switches ; prompt ; { initialFolder })

Presents the user with a standard dialog and displays all files in a directory.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
prompt	the prompt that will be displayed to tell the user which file to select
initialFolder	(optional) the file path to the folder where the dialog initially starts

Switches can be:

-SelectDefault	(Mac OS X) select the default location. the initial folder appears as the current selection
-SelectPackages	(Mac OS X) Allow selection of Mac OS X packages, like application packages.
-OpenPackages	(Mac OS X) Allow Mac OS X packages, like application packages, to be opened and navigated
-HideInvisibles	hides invisible files in the selection dialog.
-AllowMultipleFiles	allow the user to select multiple files in the dialog.

The following switch no longer works (on OS X):

-ReturnFSSpec	deprecated! the function no longer returns the folder as a FSSpec instead of a full path.

Returned result

The function returns a fullpath for the selected file to be used with one of the file manipulation functions.

If the user cancels an error code of "$$-1" is returned.

Special considerations

New: filtering on OS X is now the same as on Windows. To change the default filtering see the TrFile_SetDefaultType function, or take a look at "Filtering Files" in the example databases.

NOTE: OS X packages, like application packages, are actually folders, so this will return a path to a folder!

When the switch "-AllowMultipleFiles" is added the you can use the Command key on OS X or the Control + Alt key on Windows to select or deselect extra items. To select a contiguous group of files at once, click on the first file, then hold Shift and click the last one and all files are selected.
Each selected file is returned on a new line.

Example usage

On Mac OS X:

Set Field[MyFileName, TrFile_SelectFileDialog("-Unused" ; "Please choose a file to import" ; "Mac HD:Solution Files:Import:")]

Set Field[MyFileName, TrFile_SelectFileDialog("-SelectPackages" ; "Please choose a file to import" ; "Mac HD:Solution Files:Import:")]

Or on Windows:

Set Field[MyFileName, TrFile_SelectFileDialog("-Unused" ; "Please choose a file to import" ; "C:\Solution Files\Import\")]

This will start the selection at the folder "Import" in the "Solution Files" Folder.

TrFile_SelectFileDialog

It returns "HD Mac:Text files:My Letter" if the user selects that particular file on Mac OS. On Windows it may return "C:\Text files\My Letter".

TrFile_SelectFolderDialog

Syntax TrFile_SelectFolderDialog(switches ; prompt ; { initialFolder })

With this function the user can select a folder on disk. The function will show the user a standard dialog with the hierarchy of folders on the computer.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
prompt	the prompt that will be displayed to tell the user which folder to select
initialFolder	(optional) the path to a folder on disk that will be used as the starting point for the selection dialog

Returned result

FileSpec: The function returns a FileSpec for the selected folder. This folder can be used with other functions of the plug-in.

If the user cancels an error code of "$$-1" is returned.

Example usage

Set Field[result, TrFile_SelectFolderDialog("-Unused " ; " Please choose a folder where you want to put the files." ; " C:\Data\")]

This will present a folder selection dialog. The dialog initially shows the folder: "C:\Data\". The user can then select a folder and click on OK. The function returns with the path to the folder, for example: "D:\MyFiles\Databases\".

Example 2

TrFile_SelectFolderDialog

We assume that in your FileMaker file the following fields are defined:

gFolderPath	Global, text
gInitialFolder	Global, text

gInitialFolder should contain the path to a folder, for example "D:\Logs\Data\" (Windows) or "Mac HD:Logs:Data:" (Mac OS). In a script add the following script step:

```
Set Field[gFolderPath, TrFile_SelectFolderDialog(
        "-Unused" ; "Please choose a folder where you want to put the files." ; gInitialFolder )]
```

After the dialog gFolderPath will contain the path to the selected folder.

TrFile_SetContents

Syntax TrFile_SetContents(switches ; text)

Sets the contents of an existing file indicated by the “SetDefaultFileSpec” function.

Parameters

switches these alter the behaviour of the function
text the characters that you want to write in the file

Switches can be empty or one of this:

-Encoding=Native	(default)
-Encoding=UTF8	
-Encoding=ASCII_DOS	
-Encoding=ASCII_Windows	
-Encoding=ASCII_Mac	
-Encoding=ISO_8859_1	(Windows Latin-1)
-Encoding=BytesOnly	only characters in the range 0-255 are written, others are skipped

This determines the character encoding of the text to be written.
You can also add this switch:
-CRtoCRLF Use Windows line endings. Returns in the text parameter are substituted with CRLF (Carriage Return followed by a Line Feed) in the written file.

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The string of characters was written in the file
$$-1	genericErr	The file could not be opened for writing (You may need to create the file first).

Other errors may be returned.

Special considerations

See also the functions:
"TrFile_SaveFileDialog" to get a FileSpec for the file and "TrFile_SetDefaultFileSpec" to indicate the file to affect.

Note that the maximum size of SetContents is currently 150 million characters which equals to about max. 600 million characters written. This depends on the encoding, for example one Unicode character encoded as UTF-8 can require up to 4 characters in the written file.
Use the TrFile_AppendContents function if you need to append more characters.

The default is -Encoding=Native, which is the current encoding as set in the system preference, for example on western Windows systems it usually is ISO_8859_1 or ASCII_Windows. On Mac OS X the default will be ASCII_Mac. To be able to export all possible characters (like chinese characters) it is best to export with encoding UTF-8.

Example usage

Assume your C disk already contains a file "Testtext.txt" which is empty. We assume that a global number field gErrorCode is defined. Create the following script:

```
Set Field[gErrorCode, TrFile_SetDefaultFileSpec( "C:\Testtext.txt" )]
Set Field[gErrorCode, TrFile_SetContents( "-unused" ; "Hello there." )]
```

This script will set the contents of the file "C:\Testtext.txt" to the text "Hello there.".

Example 2

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gFilePath	Global, text
textField	text

gFilePath should contain the path to an existing file, for example "D:\Hello.TXT" (Windows) or "Disk1:Hello" (Mac).
TextField contains the text you want to write. In a script add the following scriptstep:

```
Set Field[gErrorCode, TrFile_SetDefaultFileSpec( gFilePath )]
Set Field[gErrorCode, TrFile_SetContents( "-Encoding=ASCII_Windows" ; textField )]
```

This will set the default file to your file. Then the text is written to the file.

TrFile_SetDefaultCreator

Syntax TrFile_SetDefaultCreator(switches ; Creator)

Specifies the file creator to be used when creating a new file. This creator will be used for subsequent calls to TrFile_CreateFile.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
Creator	The Creator of a file is a 4 character code used to designate the application that created a file. For example: use "ttxt" to create a SimpleText text file

Returned result

This function return an error code.

Special considerations

See also the function "TrFile_CreateFile" to create a new empty file.
Mac only. Ignored on Windows.
Starting from v3.0.5: if you use "" as creator no creator is used for newly create files.

Example usage

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gFilePath	Global, text
textField	text

gFilePath should contain the path to an existing file, for example "D:\Hello.TXT" (Windows) or "Disk1:Hello" (Mac).
TextField contains the text you want to write. In a script add the following scriptstep:

```
Set Field[gErrorCode, TrFile_SetDefaultFileSpec( gFilePath )]
Set Field[gErrorCode, TrFile_SetDefaultCreator( "ttxt" )]
Set Field[gErrorCode, TrFile_SetDefaultType( "TEXT" )]
Set Field[gErrorCode, TrFile_SetContents( "-Encoding=ASCII_Windows" ; textField )]
```

This will set the default file to your file with a SimpleText FileType and Creator. Then the text is written to the file.

TrFile_SetDefaultFileSpec

Syntax TrFile_SetDefaultFileSpec(switches ; filePath)

Indicates a FilePath (i.e. the file) to be used in succeeding functions where no FilePath is indicated.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
filePath	the path to the file

Returned result

This function does not return anything at the moment. This may change in a future version.

Special considerations

NEW you can use FileMaker styled paths, like "filewin:/C:/MyFiles/test.txt".

Example usage

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gFilePath	Global, text
textField	text

gFilePath should contain the path to an existing file, for example "D:\Hello.TXT" (Windows) or "Disk1:Hello" (Mac).
TextField contains the text you want to write. In a script add the following scriptsteps:

```
Set Field[gErrorCode, TrFile_SetDefaultFileSpec( gFilePath )]
Set Field[gErrorCode, TrFile_SetDefaultCreator( "ttxt" )]
Set Field[gErrorCode, TrFile_SetDefaultType( "TEXT" )]
Set Field[gErrorCode, TrFile_SetContents( "-Encoding=ASCII_Windows" ; textField )]
```

This will set the default file to your file with a SimpleText FileType and Creator. Then the text is written to the file.

TrFile_SetDefaultType

Syntax TrFile_SetDefaultType(switches ; fileTypes)

Sets the fileTypes to be displayed when showin gSelectFileDialog. Also sets the default file type when creating new files.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
fileTypes	determine which files are shown. Also the first will be used when creating a file.

If no fileType is specified, the TrFile_SelectFileDialog function will display all files.

The fileTypes parameter must be formatted in pairs of lines. You can specify multiple pairs of lines. The first line of the pair should contain a description of the files to be shown in the dialog. For example: "Text Files (*.txt)". This is only visible on Windows.
The second line of the pair should contain the extension to be selectable. For example: "*.txt". Also in the second line you can use multiple extensions to filter on by separating them with a semi-colon ";". For example: "*.txt;*.doc".

Returned result

This function does not return anything at the moment. This may change in a future version.

Special considerations

See also the function "TrFile_CreateFile" to create a new empty file.
New with v8.0: the format of the filter parameter is now the same on OS X and on Windows!

Example usage

This example will initially show files with the extensions .fmp12 and .fp7 (line 1 and 2). The user can change the popup to only show JPEG files (line 3 and 4) or to show all files (line 5 and 6).

```
Set Variable[ dontCare, TrFile_SetDefaultType( "-Unused ";
      "FileMaker Pro files (*.fmp12 ; *.fp7)¶" &
      "*.fmp12;*.fp7¶" &
      "JPEG (*.jpg;*.jpe;*.jpeg)¶" &
      "*.jpg;*.jpe;*.jpeg¶"  &
      "All files (*.*)¶" &
      "*.*" ) ]
Set Field[MyFileName, TrFile_SelectFileDialog( "-Unused" ; "Please choose a file"  )]
```

TIP The * means 1 or more characters. On Windows only, if you use "a*.*" you can even filter on all filenames starting with the letter a. You an also limit to a specific file name, for example "project1.doc" This will only let users select files named "project1.doc".

Example 2

This shows creating a file with a Creator and FileType, NOTE that Creator and FileType are phased out on OS X, but you can still set them. On Windows the Creator/FileType are ignored.

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gFilePath	Global, text
textField	text

TrFile_SetDefaultType

gFilePath should contain the path to an existing file, for example "D:\Hello.TXT" (Windows) or "Disk1:Hello" (OS X).
TextField contains the text you want to write. In a script add the following script steps:

```
Set Field[gErrorCode, TrFile_SetDefaultFileSpec( gFilePath )]
Set Field[gErrorCode, TrFile_SetDefaultCreator( "ttxt" )]
Set Field[gErrorCode, TrFile_SetDefaultType( "TEXT" )]
Set Field[gErrorCode, TrFile_SetContents( "-Encoding=ASCII_Windows" ; textField )]
```

This will set the default file to your file with a TextEdit FileType and Creator. Then the text is written to the file.

TrFile_SetFileAttribute

Syntax TrFile_SetFileAttribute(switches ; filePath ; cftype ; comment)

Sets an attribute of the file specified by the filePath.

Parameters

switches	these determine the attribute that is set
filePath	the path to the file
cftype	(only for creator, file type) the 4 letter Creator or File Type code
comment	(only for finder comment) the text of the comment to set

switches can be one ore more of the following:

-Lock	make the file locked (Mac OS) or Read-only (Windows)
-Unlock	make the file unlocked (Mac OS) or remove the Read-only attribute (Windows)
-Hide	make the file hidden (invisible)
-Unhide	unhide the file (visible)
-SetArchive	set the file's archive bit (Windows only)
-ResetArchive	reset the file's archive bit (Windows only)
-LabelNum=X	set the label number of a file or folder (Mac OS only)
-Findercomment	set the the finder comment of a file (Mac OS only)
-Creator	set the 4 letter Creator of a file (Mac OS only)
-Filetype	set the 4 letter FileType of a file (Mac OS only)

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The attribute was set.
$$-43	fnfErr	File not found.
$$-50	paramErr	There was an error with the parameter.
$$-1	genericErr	An unspecified error occurred.

Other errors may be returned.

Special considerations

You'll get a parameter error ($$-50) if you try to do opposite actions at the same time, like "-unlock -lock".

If you specify "0000" (4 zero's) for the cftype the creator or file type will be set to empty.

from v6.0 you can now set the label of a folder too.

Example usage

```
Set Field[result, TrFile_SetFileAttribute( "-lock" ; "C:\test.doc" )]
```

The file will be locked (made Read-Only) and the result will be 0 (provided the file exists).

```
Set Field[result, TrFile_SetFileAttribute( "-creator" ; "NELL:data:test.doc" ; "MSIE" )]
Set Field[result, TrFile_SetFileAttribute( "-filetype" ; "NELL:data:test.doc" ; "TEXT" )]
```

The file's Creator and FileType will be changed to a Internet Explorer text file.

Example 2

We assume that in your FileMaker file the following fields are defined:

gFileSpec	Global, text
gLockIt	Global, number (used as a boolean value)
gErrorCode	Global, number

TrFile_SetFileAttribute

gFileSpec should contain the path to an existing file, for example "D:\Out.txt" (Windows) or "Mac HD:Out.txt" (Mac). In a script add the following script step:

```
Set Field[gErrorCode, TrFile_SetFileAttribute(
                        If (gLockIt = 1, "-lock","-unlock") ; gFileSpec )]
```

This will make the file locked if gLockIt is 1 and unlocked otherwise.

TrFile_SetMetaData

Syntax TrFile_SetMetaData(switches ; sourceFilePath; destFilePath ; metadatablock)

Sets metadata into an image file. Metadata can contain a description of the image, copyright information, etc.

Parameters

switches these determine the metadata that is set
sourceFilePath the path to the source file
destFilePath (optional) the path to the destination file
metaDataBlock text formatted for this function

switches can be one or more of the following:
-SetIPTC set the IPTC metadata (Photoshop's File Info)
-SetExif set the Exif metadata
-MacCharSet the IPTC data will be written Mac encoded
-WinCharSet the IPTC data will be written Windows encoded
-UTF8CharSet the IPTC data will be written as UTF-8 encoded characters (new!)

Returned result

The returned result is an error code:
0 (no error) the metadata was set
$$-43 fnfErr Source file not found

Other errors may be returned.

Special considerations

If the destination parameter is empty the metadata will be saved in the source file itself. This happens also when you specify the same source file and destination file.
Currently only works with JPEGs.
ISSUES: Photoshop (or other programs) may have added other types of metadata, like XMP. These types of metadata are not changed at the moment. This may cause inconsistent metadata in the file.

If you want to remove IPTC data set the metaDataBlock to ""

See the example files "IPTC_XMP_Metadata" for information on how to create the metadata block.

Example usage

Set Field[gErrorCode, TrFile_SetMetaData("-SetIPTC" ; "C:\myData\Photo1.jpg"; ""; "##IPTC 120¶ my caption ¶##END 120")]

This will set the IPTC data, in this case only a caption. Note the returns in the IPTC data.

Example 2

Set Field[gErrorCode,
TrFile_SetMetaData("-SetIPTC " & this::IPTC_DestinationCharSet ; this::PathToSourceFile; this::PathToDestinationFile ; this::gMetaDataBlock)]

gMetaDataBlock should contain something similar to this below:

```
##IPTC 120
FOR IMMEDIATE RELEASE--FILE--Shania Twain performs at the Country Music Association Awards show in Nashville, Tenn., ...
##END 120
##IPTC 122
```

TrFile_SetMetaData

CJC RWP MAH
##END 122
##IPTC 105
SHANIA TWAIN
##END 105
##IPTC 40
FOR IMMEDIATE RELEASE. A SEPT. 22, 1999 FILE PHOTO. DIGITAL IMAGE
##END 40
...

TrFile_SetTimestampCreated

Syntax TrFile_SetTimestampCreated(switches ; filePath ; timestamp)

Sets the creation date/time of the file specified by the filePath.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
filePath	the path to the file
timestamp	new creation date/time of the file

Returned result

If successful it returns 0.
If unsuccessful it returns an error code starting with $$ and the error code. Possible codes are:

$$-43	The file was not found
$$-4241	The date/time is too big
$$-4245	The date/time is too small

Other error codes might be returned.

Special considerations

- Mac OS X limits (using file system HFS+): on Mac OS X the minimum representable date is January 1st, 1904 GMT. The maximum representable date is February 6, 2040 at 06:28:15 GMT.
- Windows limits: The minimum representable date is January 1st, 1980. The maximum date is December 31, 2107.
- The timestamps are in local time.
- On Mac OS X it may take some time (about 5 secs) before the change is visible in the Finder. On Windows you need to refresh a Window before you see the change.
- Some file systems may not support creation time.

Example usage

```
Set Field[result, TrFile_SetTimestampCreated( "-Unused" ; "C:\test.doc" ;
                                    Timestamp ( Date ( 6 ; 30 ; 2005 ) ; Time ( 10 ; 59 ; 59 ) ))]
```

This will set the creation date/time to June 30th, 2005 at 10:59:59.

Example 2

We assume that in your FileMaker file the following fields are defined:

gFileSpec	Global, text
gTimestampCreated	Global, timestamp
gErrorCode	Global, text

gFileSpec should contain the path to an existing file, for example "D:\Out.txt" (Windows) or "Mac HD:Out.txt" (Mac).
gTimestampCreated should contain a valid date/time. In a script add the following script step:

```
Set Field[gErrorCode, TrFile_SetTimestampCreated( "-Unused" ;  this::gTheFile ; this::gTimestampCreated )
```

This will set the file's creation date/time to the value in the timestamp field.

TrFile_SetTimestampModified

Syntax TrFile_SetTimestampModified(switches ; filePath ; timestamp)

Sets the modification date/time of the file specified by the filePath.

Parameters

switches	reserved for future use, leave empty or set to "-Unused"
filePath	the path to the file
timestamp	new modification date/time of the file

Returned result

If successful it returns 0.
If unsuccessful it returns an error code starting with $$ and the error code. Possible codes are:

$$-43	The file was not found
$$-4241	The date/time is too big
$$-4245	The date/time is too small

Other error codes might be returned.

Special considerations

- Mac OS X limits (using filesystem HFS+): on Mac OS X the minimum representable date is January 1st, 1904 GMT. The maximum representable date is February 6, 2040 at 06:28:15 GMT.
- Windows limits: the minimum representable date is January 1st, 1980. The maximum date is December 31, 2107.
- The timestamps are in local time.
- On Mac OS X it may take some time (about 5 secs) before the change is visible in the Finder. On Windows you need to refresh a Window before you see the change.

Example usage

```
Set Field[result, TrFile_SetTimestampModified( "-Unused" ; "Mac HD:data:test.txt" ;
                                    Timestamp ( Date ( 6 ; 30 ; 2005 ) ; Time ( 10 ; 59 ; 59 ) ))]
```

This will set the modification date/time to June 30th, 2005 at 10:59:59.

Example 2

We assume that in your FileMaker file the following fields are defined:

gFileSpec	Global, text
gTimestampCreated	Global, timestamp
gErrorCode	Global, text

gFileSpec should contain the path to an existing file, for example "D:\Out.txt" (Windows) or "Mac HD:Out.txt" (Mac).
gTimestampCreated should contain a valid date/time. In a script add the following script step:

```
Set Field[gErrorCode, TrFile_SetTimestampModified( "-Unused" ;  this::gTheFile ; this::gTimestampCreated )
```

This will set the file's modification date/time to the value in the timestamp field.

TrFile_Substitute

Syntax TrFile_Substitute(switches ; sourceFile ; destinationFile ; search string ; replace string)

Substitutes every occurrence of a specified set of characters in a (text) file with another set of characters you specify, and writes the revised file. The Substitute function is case sensitive.

Parameters

switches	these determine how the function works. At the moment you can specify how the function handles case differences
sourceFile	the path to the file which is the source for the substitution
destinationFile	the path where the resulting file must be written to
search string	any text expression or field containing text
replace string	any text expression or field containing text

switches can be one of the following:

-CaseSensitive	substitute strings only if the case of the source string is the same
-IgnoreCase	substitute strings even if the case differs

Switches can also be one of this:

-Encoding=Native	(default)
-Encoding=UTF8	
-Encoding=ASCII_DOS	
-Encoding=ASCII_Windows	
-Encoding=ASCII_Mac	

This determines the character encoding of the text to be read.

Returned result

The returned result is an error code. An error always starts with two dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The destination file with the substitutions was created.
$$-43	fnfErr	Source file not found, or destination directory does not exist
$$-48	dupFNErr	Destination file already exists
$$-1	genericErr	The substitution could not be made.

Other errors may be returned.

Special considerations

If the destination parameter is empty the substitution result will be saved in the source file itself. This happens also when you specify the same source file and destination file (this possibility was added in version 2.5). Note that the plug-in uses a temporary file to implement this feature, which can require the same amount of disk space as the source file. The plug-in can fail if there is not enough space on the disk.

NOTE The length of the search and replace string can be up to 1 GB.

Up to v5.0 there was an error in the documentation: it was incorrectly stated that UTF-8 was the default encoding. The default is -Encoding=Native, which is the current encoding as set in the system preference, for example on western Windows systems it usually is ISO_8859_1 or ASCII_Windows.

Example usage

```
Set Field[result, TrFile_Substitute(
                "-IgnoreCase" ; "C:\Data\temp.tab" ; "C:\Main.tab" ; "country" ; "world" )]
```

This will create the file "C:\Main.TAB" where all occurences of "country" are replaced with the string "world". The case is ignored, so that also a string "Country" will be substituted.

TrFile_Substitute

Example 2

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number	gSearchStr	Global, text
gSourceFilePath	Global, text	gReplaceStr	Global, text
gDestFilePath	Global, text		

gSourceFilePath should contain the path to an existing file, for example "D:\TempLogs\Log01.TXT" (Windows) or "Mac HD:TempLogs:Log 1" (Mac).
gDestFilePath should contain the path to the destination and should not exist, for example "D:\Output\L2000_01.TXT" (Windows) or "Mac HD:Output:Log 2000_01" (Mac). In a script add the following scriptstep:

```
Set Field[gErrorCode, TrFile_Substitute( "-CaseSensitive" ; gSourceFilePath ; gDestFilePath ;
                                         gSearchStr ; gReplaceStr )]
```

This will create a new file gDestFilePath where all occurences of gSearchStr are replaced with the contents of gReplaceStr.

TrFile_UnmountDisk

Syntax TrFile_UnmountDisk(switches ; diskname)

Unmounts and ejects a remote or removable disk.

Parameters

switches	not used, reserved for future use. Leave blank or put "-Unused"
diskname	the name of the disk to be unmounted or ejected

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The disk was ejected.
$$-35	nsVolErr	No such volume (Wrong disk name or not mounted).
$$-47	fBsyErr	The disk is in use.
$$-5010	afpFileBusy	Can't eject as the disk is being shared.

Other errors may be returned.

Special considerations

Best to use only on Ejectable Disks or remote servers.

Since version 2.6 this function is also available on Windows. However it does not eject removable media, like a CD-ROM.

Example usage

Set Field[result, TrFile_UnmountDisk("Mac HD:")]

This will eject the hard disk "Mac HD", provided it is not the startup disk. On Windows an example is:

Set Field[result, TrFile_UnmountDisk("K:")]

This will unmount the mapped drive "K:"

Example 2

We assume that in your FileMaker file the following fields are defined:

gErrorCode	Global, number
gDiskName	Global, text

gFilePath should contain the path to the disk, for example "Archive CD:" (Mac). In a script add the following script step, which will eject the disk specified by gDiskName:

```
Set Field[gErrorCode, TrFile_UnmountDisk( gDiskName )]
```

TrFile_UnZip

Syntax TrFile_UnZip(switches ; sourceZipFileSpec ; destFolderSpec)

Expands a ZIP file into the specified folder.

Parameters

switches	modifies the behavior of the function
sourceZipFileSpec	the path to the ZIP file to be unzipped
destFolderSpec	the path to the folder where the ZIP is going to be expanded

You can add one or more of these switches to retrieve extra information:

-DontCreateArchiveFolder	don't create an extra top folder if there are multiple items at the top level of the ZIP file
-OverwriteExisting	overwrite existing items in the destination folder(s)

Returned result

The returned result is an error code. An error always starts with 2 dollars, followed by the error code. You should always check for errors. Returned error codes can be:

0	no error	The ZIP file was unzipped
$$-43	fnfErr	Source file not found
$$-48	dupFNErr	Destination ZIP folder already exists

$$-4230 timeoutErr The extracting of the ZIP timed out.
$$-4281 kErrUnZIPFileExtractConflict A folder already exists with the same name as a file to be extracted.
$$-4282 kErrUnZIPFolderExtractConflict A file already exists with the same name as a folder to be extracted.

Other errors may be returned.

Special considerations

The TrFile_UnZip function will create the complete (sub)folder structure of the ZIP. This may take a bit of time for large ZIP files.

WARNING only use the switch "-OverwriteExisting" when you are sure you don't overwrite something important. If you first want to see what is in the ZIP file you can use the TrFile_GetZipInfo function to get a list of items. Alternatively it might be better to extract the contents of the ZIP in an empty folder first, and then copy the extracted files from that folder.

Example usage

Set Field[$ErrorCode ; TrFile_UnZip("-Unused" ; "D:\myZIP2016.zip"; "C:\resultDir\")]

This will uncompress the ZIP file myZIP2016.zip into the folder resultDir. For example these files and folders might be created on the disk:

C:\resultDir\my report\illustration.jpg
C:\resultDir\my report\my report\extra data\
C:\resultDir\my report\my report\extra data\note.txt

Example 2

TrFile_UnZip

We assume that in your FileMaker file the following fields are defined and filled with the paths to the source ZIP file and destination folder:

this::sourceZipPath may contain something like "MacHD:Users:kip:bigtext.zip"
this::destOfUnZIP_Path may contain something like "MacHD:Users:destFolder"

This example will first set the variables from the paths stored in the two fields, and then expand the ZIP file. In a script add the following steps:

```
#Initialize: copy the fields into variables first:
Set Variable [ $SourceZIPfilePath;  this::sourceZipPath ]
Set Variable [ $DestinationFolder;  this::destOfUnZIP_Path ]

#Now unZIP it:
Set Variable [ $ErrorCode;  TrFile_UnZip( "-OverwriteExisting"  ; $SourceZIPfilePath ; $DestinationFolder) ]

#Return the resulting error code:
Exit Script [ $ErrorCode ]
```

NOTE: in this example we have specified the switch "-OverwriteExisting". Only use this switch if you want to allow that existing items in the destination folder can be overwritten.

TrFile_Version

Syntax TrFile_Version(switches)

Use this function to see which version of the plug-in is loaded.
Note: This function is also used to register the plug-in.

Parameters

switches determine the behaviour of the function

switches can be one of this:

-GetVersionString	the version string is returned (default)
-GetVersionNumber	returns the version number of the plug-in
-ShowFlashDialog	shows the Flash Dialog of the plug-in (returns 0)
-GetPluginInstallPath	returns the path where the plug-in is installed
-GetRegistrationState	get the registration state of the plug-in: 0 = not registered ; 1 = registered
-UnregisterPlugin	sets the registration state of the plug-in to unregistered

If you leave the parameter empty the version string is returned.

Returned result

The function returns ? if this plug-in is not loaded. If the plug-in is loaded the result depends on the input parameter. It is either a:

VersionString:
If you asked for the version string it will return for example "Troi File Plug-in 4.6"

VersionNumber:
If you asked for the version number it returns the version number of the plug-in x 1000. For example version 4.6 will return number 4600.

-ShowFlashDialogResult:
This will show the flash dialog and then return the error code 0.

-GetRegistrationState:
returns 0 = the plug-in is not registered ; 1 = the plug-in is registered.

Special considerations

IMPORTANT always use this function to determine if the plug-in is loaded. If the plug-in is not loaded use of external functions may result in data loss, as FileMaker will return an empty field to any external function that is not loaded.

Example usage

We assume that a calculation number field cVersion is defined like this:

```
cVersion = TrFile_Version
```

This will evaluate to "Troi File Plug-in <version number>". For version 6.1 this will be: "Troi File Plug-in 6.1".

Example 2

TrFile_Version("-GetVersionNumber") will return 4600 for version 4.6
TrFile_Version("-GetVersionNumber") will return 4620 for version 4.6.2

So for example to use a feature introduced with version 4.6 test if the result is equal or greater than 4600.

TrFile_VersionAutoUpdate

Syntax TrFile_VersionAutoUpdate

Use this function to see which version of the plug-in is loaded, formatted for FileMaker Server's AutoUpdate function. Returns 8 digit number to represent an AutoUpdate version.

Parameters

none

Returned result

The function returns ? if this plug-in is not loaded. If the plug-in is loaded the result is a version number, it is returned in the format aabbccdd where every letter represents a digit of the level, so versions can be easily compared.

Special considerations

The TrFile_VersionAutoUpdate function is part of an emerging standard for FileMaker plug-ins of third party vendors of plug-ins. The version number can be easily compared, when using the Autoupdate functionality of FileMaker Server.

Example usage

TrFile_VersionAutoUpdate will return 04060000 for version 4.6
TrFile_VersionAutoUpdate will return 04060203 for version 4.6.2.3

So for example to use a feature introduced with version 4.6 test if the result is equal or greater than 04060000.