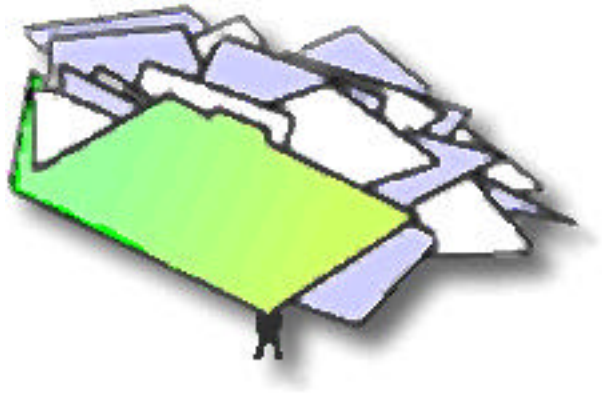


TROI FILE PLUG-IN™ 2.1.2 USER GUIDE

September 2000

Troi Automatisering
Vuurlaan 18
2408 NB Alphen a/d Rijn
The Netherlands
Tel: +31 172-426606 Fax: +31 172-470539

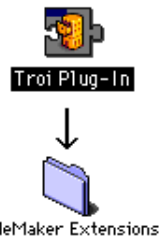
You can also visit the Troi web site at: <http://www.troi.com/> for additional information.

Table of Contents

Installing plug-ins

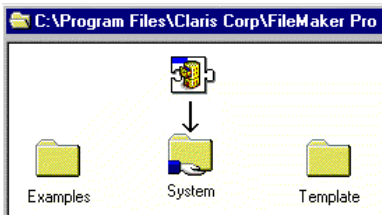
For Macintosh:

- Quit FileMaker Pro.
- Put the file "Troi File Plug-in" from the folder "MacOS Plug-in" into the "FileMaker 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 Plug-In” 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 "trfile.fmx" from the directory "Windows" into the "System" sub-directory 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 'file.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 "File Example.fp3" to see how to use the plug-in's functions. There is also a Function overview available.

IMPORTANT There is a problem in FileMaker Pro 4.0v1. Please make sure that all plug-ins that are in the folder "FileMaker Extensions" are enabled in the preferences. (Under Edit/ Preferences/ Application/ Plug-ins). Make sure all plug-ins have a cross before their name. Remove plug-ins you don't use from the "FileMaker Extensions" folder. This bug is fixed in version 4.0v2, 4.1 and 5.0.

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. 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.

If you find any mistake 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!

What can this plug-in do?

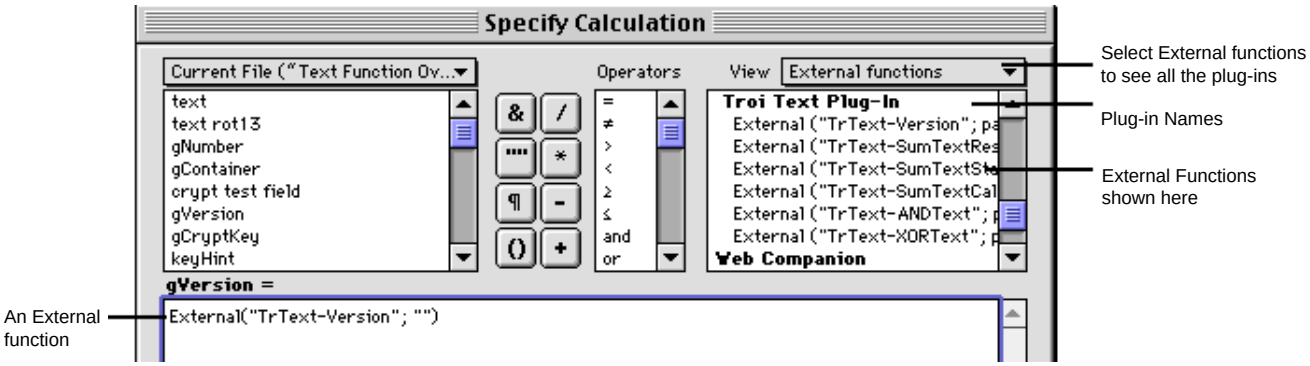
The Troi File Plug-in is a very powerful tool for getting access to information outside the FileMaker database. Any files stored on the rest of the computer can be accessed through the functions of the plug-in. All from within FileMaker you can:

- 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

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: External("function name", parameter) where function name is the name of an external function. The parameter is required, even if it's only "". Plug-ins don't work directly after installation. To access a plug-in function, you need to add the calls to the function in a ScriptMaker Script.

IMPORTANT In the United States, commas act as list separators in functions. In other countries semicolons might be used as list separators. The separator being used depends on the operating system your computer uses, as well as the separator used when the file was created. All examples show the functions with commas.

Where to add the External Functions?

External functions for this plug-in are intended to be used in a script step using a calculation. For most functions of this plug-in it makes no sense to add them to a define field calculation, as the functions will have side effects.

About FileSpec's

To be able to specify a file or folder, the plug-in uses FileSpec, which is short for File Specification.

On Windows this 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\".

On Mac OS a FileSpec can either be a full path or an FSSpec. A full path looks like this:
"Mac HD:Data Files:Database:testfile"
and similar to this for a folder:
"Archive CD:My Files:Main:"

Alternatively FileSpec can also be a FSSpec, which has the following format:
":volumeID:directoryID:fileName"

The plug-in recognizes the FSSpec by the colon as the first character.
An example may be: ":-1:300:testfile".

NOTE On Mac OS a volume name and therefore a full path need not be unique. FSSpec are always 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". In ScriptMaker create a script "Get Report Text". Add the following script step to this script:

```
Set Field[myTextField, External("TrFile-GetContents", "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 case sensitive. Be sure to spell them right, or get them from the External Functions list at the top right of the "Specify Calculation" dialog.

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 global text field, which holds the path to the folder. Say the full path on Mac OS would be "Disk:DataFiles:Report.txt". Also assume you have defined a global text field "gPathToFolder". The script would then become:

```
If [Status(CurrentPlatform) = 1]
    Comment[ ...on Mac OS...]
    Set Field[gPathToFolder, "Disk:DataFiles:")]
Else
    Comment[ ...on Windows...]
    Set Field[gPathToFolder, "C:\DataFiles\")]
Endif
Set Field[myTextField, External("TrFile-GetContents", 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 42 functions of the plug-in as building blocks. Together they give you all the tools you need to access the disk.

Summary of functions

The Troi File Plug-in adds the following functions:

function name	short description
TrFile-AppendContents	append characters to the contents of an existing file.
TrFile-ContentsDialog	shows the user a file selection dialog.
TrFile-Control	controls (disable and enable)the functions of the plug-in.
TrFile-CopyFile	copies a file to the specified path.
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-CreateFile	creates a new empty file.
TrFile-CreateFolder	creates a new folder.
TrFile-DeleteFile	deletes a file indicated.
TrFile-DeleteFolder	Deletes the folder indicated by the FolderSpec.
TrFile-DiskInfo	Retrieves information about a specified disk.
TrFile-FileSpec To FullPath	get full path of a file or folder from an FSSpec.
TrFile-FindFolder	finds special folders.
TrFile-FolderList	lists the contents of a folder.
TrFile-FullPath To FileSpec	changes the full name path to the standard Mac OS FSSpec.
TrFile-Get FileSpec Dialog	presents the user with a standard dialog and displays all files in a directory.
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-GetDateTimeCreated	get the creation date and time of a file.
TrFile-GetDateTimeModified	get the modification date and time of a file.
TrFile-GetFileAttribute	returns an attribute of the file specified by the FileSpec.
TrFile-GetFileCreator	get the Creator of a file.
TrFile-GetFileSize	get the file size of a file.
TrFile-GetFileType	get the FileType of a file.
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-Launch	opens a file with the file's application.
TrFile-ListDisks	lists the names of all currently available disks.
TrFile-MetaData	gets metadata out of a graphic file.

TrFile-MoveFile	moves (or renames) a file from one disk location to another.
TrFile-ReferenceToClip	imports picture and store only a reference to it.
TrFile-Save FileSpec Dialog	presents the user with a save file dialog.
TrFile-Search	searches a volume (disk) for a file or folder (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-Substitute	substitutes characters in a (text) file.
TrFile-ThumbnailToClip	creates a thumbnail from a graphics file and puts it on the clipboard.
TrFile-Version	use this function to see which version of the plug-in is loaded.

Function Reference
TrFile-AppendContents

Syntax Set Field [result, External("TrFile-AppendContents", "text")]

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

Parameters

text: the characters that you want to add at the end of the default 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).

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.

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, External("TrFile-SetDefaultFileSpec", "C:\Testtext.txt" )]
If [gErrorCode = 0]
        Set Field[gErrorCode, External("TrFile-AppendContents", " there." )]
Endif
```

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\L2000_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. In ScriptMaker add the following scriptstep to your startup script:

TrFile-AppendContents

Set Field[gErrorCode, External("TrFile-SetDefaultFileSpec", 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, External("TrFile-AppendContents",
        DateToText(Status(CurrentDate)) & " " &
        TimeToText(Status(CurrentTime)) & " text: " & gLogtext & gXplatformReturn))]
```

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

12/31/2000 10:23:56 text: user peter logged in
12/31/2000 14:31:02 text: user peter logged out
12/31/2000 14:31:02 text: closed file "Invoices.fp5"
etc...

TrFile-ContentsDialog

Syntax Set Field [result, External("TrFile-ContentsDialog", "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

prompt: *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.*

Returned result

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

Special Considerations

FileMaker fields are limited to 64000 characters. So files longer than 64000 bytes are cut after the 64000th character.
Use the partial reading of "TrFile-GetContents" to read in longer files.
Aslo note that high ASCII characters are not converted to the internal Filemaker format. Use "TrFile-ConvertToFMText" to convert to the Filemaker format.

Example Usage

```
Set Field [ result, External["TrFile-ContentsDialog", "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, External["TrFile-ContentsDialog", "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 Set Field [result, External("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 acces to functions*
-enableAllFunctions *allow the user acces to all functions*

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.

Example Usage

```
Set Field[result, External("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". In ScriptMaker add the following to the startup script:

```
Set Field[gErrorCode, External("TrFile-Control",  "-disableAllFunctions|" & gPassword)]
```

TrFile-Control

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

```
Set Field[gErrorCode, External("TrFile-Control",  "-enableAllFunctions|" & gPassword)]
... add powerful functions here...
Set Field[gErrorCode, External("TrFile-Control",  "-disableAllFunctions|" & gPassword)]
```

This will temporary enable the plug-in functions.

TrFile-ConvertFromFMText

Syntax Set Field [result, External("TrFile-ConvertFromFMText", "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

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

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>.

Note: For long texts the result can be longer than the maximum number of characters (64000). In this case the function will truncate the length to 64000 characters.

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 scritstep:

```
Set Field[result, External("TrFile-ConvertFromFMText", "Bjørn, ¶Münchenstraße 2")]
```

The result will look like:
"Bj¯rn,
M¸nchenstrafle 2".

This may look wrong, but when you use for exampe the SetContents function it will appear correct on 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 ScriptMaker add the following script steps:

```
Set Field[gContentsConverted, External("TrFile-ConvertFromFMText", contents)]
```

TrFile-ConvertFromFMText

```
## now gContentsConverted is in the format of the operating system
 Set Field[gErrorCode,  External("TrFile-CreateFile", gOutputFile)”]
 Set Field[gErrorCode,  External("TrFile-SetDefaultFileSpec", gOutputFile)]
 If [gErrorCode = 0 ]
    Set Field[gErrorCode, External("TrFile-SetContents", gContentsConverted)]
 Endif
```

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 transparant on both Mac OS and Windows.

TrFile-ConvertToFMText

Syntax Set Field [result, External("TrFile-ConvertToFMText", "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

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

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 scritstep:

```
Set Field[result, External("TrFile-ConvertToFMText", External("TrFile-GetContents", "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 ScriptMaker add the following script step:

```
Set Field[contents, External("TrFile-GetContents", gFileSpec)]
## now the contents field is in the format of the operating system
Set Field[contents, External("TrFile-ConvertToFMText", 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 Set Field [result, External("TrFile-CopyFile", "source FileSpec | destination FileSpec")]

Copies a file to the specified path.

Parameters

source FileSpec: *the Filespec or path to the file to copy.*
destination FileSpec: *the Filespec or 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
$$-1	genericErr	The file could not be copied

Other errors may be returned.

Special Considerations

Use the TrFile-DeleteFile to delete a file first if it exists. See also the function "TrFile-Save FileSpec Dialog" 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. In ScriptMaker create the following script:

```
Set Field[gErrorCode, External("TrFile-CopyFile", "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 ScriptMaker add the following scriptstep:

```
Set Field[gErrorCode, External("TrFile-CopyFile", gSourceFilePath & "|" & gDestFilePath )]
```

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

TrFile-CreateFile

Syntax Set Field [result, External("TrFile-CreateFile", "FileSpec")]

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

Parameters

FileSpec: *the Filespec or 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

See also the function "TrFile-Save FileSpec Dialog" to get a FileSpec for the file.

Example Usage

Set Field[gErrorCode, External("TrFile-CreateFile", "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 ScriptMaker add the following scriptstep:

Set Field[gErrorCode, External("TrFile-CreateFile", gDestFilePath)]

This will create the file indicated in the gDestFilePath.

TrFile-CreateFolder

Syntax Set Field [result, External("TrFile-CreateFolder", "FileSpec")]

Creates a new (empty) folder (subdirectory).

Parameters

FileSpec : 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

See also the functions: "TrFile-Save FileSpec Dialog" to get a FileSpec for a folder,
"TrFile-CreateFile" to create a new empty file.

Example Usage

We assume that a global number field gErrorCode is defined. In ScriptMaker create the following script:

```
Set Field[gErrorCode, External("TrFile-CreateFolder", "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, External("TrFile-CreateFolder", "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 ScriptMaker add the following scriptstep:

```
Set Field[gErrorCode, External("TrFile-CreateFolder", gFolderPath )]
```

This will create the folder specified in the gFolderPath.

TrFile-DeleteFile

Syntax Set Field [result, External("TrFile-DeleteFile", "FileSpec")]

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

Parameters

FileSpec: *the Filespec or 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
$$-43	fnfErr	The file was not found
$$-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 “TrDl-DoDialog” or use the “Show Message” script step if you want to warn the user.

Example Usage

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

```
Set Field[gErrorCode, External("TrFile-DeleteFile", "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 ScriptMaker add the following scriptstep:

```
Set Field[gErrorCode, External("TrFile-DeleteFile", gFilePath )]
```

This will delete the file from the disk.

TrFile-DeleteFolder

Syntax Set Field [result, External("TrFile-DeleteFolder", "Switches |FolderSpec")]

Deletes the folder indicated by the FolderSpec.

Parameters

Switches: *determines which folders are deleted.*
FolderSpec: *the FolderSpec or path to the folder to delete.*

Switches can be empty or this:
-DeleteAllSubFoldersAndAllContents

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
$$-47	fBsyErr	File is busy, cant delete because it is not empty
$$-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 “TrDl-DoDialog” or use the “Show Message” script step if you want to warn the user.

Example Usage

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

```
Set Field[result, External("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 ScriptMaker add the following scriptstep:

```
Set Field[gErrorCode, External("TrFile-DeleteFolder", "deleteAllSubFoldersAndAllContents |" & gFolderPath )]
```

TrFile-DeleteFolder

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 Set Field [result, External("TrFile-DiskInfo", "Switch |DiskName")]

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

Parameters

Switch: *this determines which information is returned.*
Diskname: *the name of the disk.*

Switch must be one of this:
-GetFileCount
-GetFolderCount

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

Other errors may be returned.

Example Usage

Set Field[gErrorCode, External("TrFile-DiskInfo", " -GetFileCount |Mac HD")]

This command will return the number of files for of the disk "Mac HD".

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 ScriptMaker add the following script step:

Set Field[gFolderCount, External("TrFile-DiskInfo", " -GetFolderCount |" & _gDiskName)]

This will return the number of folders on the disk.

TrFile-FileSpec To FullPath

Syntax Set Field [result, External("TrFile-FileSpec To FullPath", "FileSpec")]

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

Parameters

FileSpec: *the FSSpec of a file or folder. It is of the form ":volumeID:directoryID:fileName"*

Returned result

Returns a full path formatted like this:
"DiskName:foldername1:foldername2:...:foldernameN:filename"

Special Considerations

•On the Mac a disk name need not be unique, be careful with that.•
•See also the function “TrFile-FullPath To FileSpec” to revert the path name.•

Example Usage

Set Field[result, External("TrFile-FileSpec To FullPath", ":-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, External("TrFile-FileSpec To FullPath", gFilePath)]

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

TrFile-FindFolder

Syntax Set Field [result, External("TrFile-FindFolder", "folderconstant")]

Finds special folders (subdirectories). Currently Mac OS only.

Parameters

folderconstant: Specifies which folder to return, must be one of the following:

desktop	*= desktop folder*
system	*= system folder*
trash	*= trash folder on the desktop*
shutdown	*= shutdown items folder in the system folder*
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*
temporary	*= hidden temporary folder on the startup disk*
root	*= top folder on the startup disk*

Returned result

FileSpec: The Filespec of the folder.

Special Considerations

Currently Mac OS only. Returns $$-1 on Windows.

Example Usage

Set Field [result, External["TrFile-FindFolder", "trash")]

This will return the FSSpec of the Trash folder.

TIP: With this folder and the MoveFile function you can move items into the Trash.

Example 2

Set Field [gStartupDiskFolder, External["TrFile-FindFolder", "root")]
This will return the top folder on the startup disk.

TrFile-FullPath To FileSpec

Syntax Set Field [result, External("TrFile-FullPath To FileSpec", "FileSpec")]

Changes the full name path to the standard Macintosh FSSpec.

Parameters

FileSpec: *the Full path of a file or folder. It is of the form "DiskName:foldername1:foldername2:...:foldernameN:filename"*

Returned result

Returns a Mac OS FSSpec formatted like this:
":volumeID:directoryID:fileName"

This is the preferred way on the Mac to specify a file.

Special Considerations

An FSSpec is recognized by the colon as the first character.
•On the Mac a disk name need not be unique, be careful with that.•

Example Usage

Set Field[result, External("TrFile-FullPath To FileSpec", "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 ScriptMaker add the following scriptstep:

Set Field[gFilePath, External("TrFile-FullPath To FileSpec", gFilePath)]

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

TrFile-Get FileSpec Dialog

Syntax Set Field [result, External("TrFile-Get FileSpec Dialog", "prompt | initialFolder")]

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

Parameters

prompt: *the prompt that will be displayed to tell the user which file to select.*
initialFolder: *(optional) the FileSpec or path to the folder where the dialog initially starts.*

Returned result

FileSpec: The function returns a FileSpec 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.

Example Usage

On MacOS:
Set Field[MyFileName, External("TrFile-Get FileSpec Dialog", "Please choose a file to import|Mac HD:Solution Files:Import:")]

Or on Windows:
Set Field[MyFileName, External("TrFile-Get FileSpec Dialog", "Please choose a file to import|C:\Solution Files\Import\")]

This will start the selection at the folder "Import" in the "Solution Files" Folder. It returns "HD Mac:Text files:My Letter" if the user selects that particular file on MacOS. On Windows it may return "C:\Text files\My Letter".

TrFile-GetContents

Syntax Set Field [result, External("TrFile-GetContents", "fileSpec| start | size")]

Returns (a part of) the contents of the file specified by the FileSpec. When a start is given this function extracts characters from the specified file, starting at the position specified by start and containing the number of characters specified by size.

Parameters

fileSpec:	*the Filespec or path to the file.*
start:	*(optional) starting position in the file.*
size:	*(optional) the number of characters to get.*

Returned result

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

Special Considerations

FileMaker fields are limited to 64000 characters. So results longer than 64000 bytes are cut after the 64000th character. Use start and size parameter to read in longer files.
Also note that High ASCII characters are not converted to the internal Filemaker format. Use "TrFile-ConvertToFMText" to convert to the Filemaker format.

Example Usage

Set Field[MyTextField, External("TrFile-GetContents", "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

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 ScriptMaker add the following scriptstep:

Set Field[gContents, External("TrFile-GetContents", gFileSpec & "|" & gStart & "|" & gLength)]

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

TrFile-GetDataSize

Syntax Set Field [result, External("TrFile-GetDataSize", "FileSpec")]

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

Parameters

FileSpec: *the Filespec or 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".

Example Usage

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

```
Set Field[MyTextField, External("TrFile-GetDataSize", "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 ScriptMaker add the following scriptstep:

```
Set Field[gSize, External("TrFile-GetDataSize", gFileSpec)]
```

This will fill gSize with the size of the data.

TrFile-GetDateCreated

Syntax Set Field [result, External("TrFile-GetDateCreated", "FileSpec")]

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

Parameters

FileSpec: *the Filespec or 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.

Example Usage

Set Field[creationDate, External("TrFile-GetDateCreated", "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 ScriptMaker add the following script step:

Set Field[creationDate, External("TrFile-GetDateCreated", gFilePath)]

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

TrFile-GetDateModified

Syntax Set Field [result, External("TrFile-GetDateModified", "FileSpec")]

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

Parameters

FileSpec: *the Filespec or 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.

Example Usage

Set Field[modificationDate, External("TrFile-GetDateModified", "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 ScriptMaker add the following script step:

Set Field[modificationDate, External("TrFile-GetDateModified", gFilePath)]

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

TrFile-GetDateTimeCreated

Syntax Set Field [result, External("TrFile-GetDateTimeCreated", "FileSpec")]

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

Parameters

FileSpec: *the Filespec or 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.

Example Usage

Set Field[result, External("TrFile-GetDateTimeModified", "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 ScriptMaker add the following script step:

Set Field[creationDateTime, External("TrFile-GetDateTimeCreated", gFilePath)]

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

TrFile-GetDateTimeModified

Syntax Set Field [result, External("TrFile-GetDateTimeModified", "FileSpec")]

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

Parameters

FileSpec: *the Filespec or 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.

Example Usage

Set Field[result, External("TrFile-GetDateTimeModified", "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 ScriptMaker add the following script step:

Set Field[modificationDateTime, External("TrFile-GetDateTimeModified", gFilePath)]

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

TrFile-GetFileAttribute

Syntax Set Field [result, External("TrFile-GetFileAttribute", "switch | FileSpec")]

Returns an attribute of the file specified by the FileSpec.

Parameters

switch: *this determines the attribute that is returned.*
FileSpec: *the FileSpec or path to the file.*

switch 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)*

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.

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.

Example Usage

```
Set Field[result, External("TrFile-GetFileAttribute", "-lockAttr| C:\test.doc")]
```

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:

TrFile-GetFileAttribute

Set Field[locked, External("TrFile-GetFileAttribute", "-lockAttr|" & fileSpec)]

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

TrFile-GetFileCreator

Syntax Set Field [result, External("TrFile-GetFileCreator", "FileSpec")]

Returns the Creator for the file specified by the FileSpec.

Parameters

FileSpec: *the Filespec or 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 4 creates database files with a Creator "FMP3".

Special Considerations

•This function is not available in Windows.•
See also the function "TrFile-GetFileType" to get the FileType.

Example Usage

Set Field[result, External("TrFile-GetFileType", "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 ScriptMaker add the following script step:

Set Field[creator, External("TrFile-GetFileType", gFilePath)]

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

TrFile-GetFileSize

Syntax Set Field [result, External("TrFile-GetFileSize", "FileSpec")]

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

Parameters

FileSpec: *the Filespec or path to the file.*

Returned result

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

Example Usage

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

```
Set Field[MyTextField, External("TrFile-GetFileSize", "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 ScriptMaker add the following scriptstep:

```
Set Field[gSize, External("TrFile-GetFileSize", gFileSpec)]
```

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

TrFile-GetFileType

Syntax Set Field [result, External("TrFile-GetFileType", "FileSpec")]

Returns the FileType for the file specified by the FileSpec.

Parameters

FileSpec: *the Filespec or 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 4 files have a FileType of "FMP3" and Applications have FileType of "APPL".

Special Considerations

•This function is not available in Windows. Windows uses the 3 letter extension, like .fp3 instead.•

Example Usage

Set Field[result, External("TrFile-GetFileType", "KES:test.fp5")]

This will (probably) return "FMP5" which is the FileType for FileMaker Pro 5.

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 ScriptMaker add the following script step:

Set Field[fileType, External("TrFile-GetFileType", gFilePath)]

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

TrFile-GetPathTo

Syntax Set Field [result, External("TrFile-GetPathTo", "switch|desiredFilename|fileSize")]

Returns the path to an object, at the moment the path to the currentFile.

Parameters

switch:	*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:

-currentFileName	*return the FileSpec of the file that is currently active (the front window).*

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:

0	no error	The attribute was set.
$$-43	fnfErr	File not found.
$$-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

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.

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).

Example Usage

```
Set Field [result, External("TrFile-GetPathTo",
        "-currentFileName |" & Status(CurrentFileName) &"|"& Status(CurrentFileSize))]
```

TrFile-GetResForkSize

Syntax Set Field [result, External("TrFile-GetResForkSize", "FileSpec")]

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

Parameters

FileSpec: *the Filespec or 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" with the text "Hello!" in it. Then:

```
Set Field[MyTextField, External("TrFile-GetResForkSize", "Disk:Read me")]
```

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 "D:\Readme.txt" (Windows) or "Mac HD:Readme.txt" (Mac). In ScriptMaker add the following scriptstep:

```
Set Field[gSize, External("TrFile-GetResForkSize", gFileSpec)]
```

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

TrFile-GetTimeCreated

Syntax Set Field [result, External("TrFile-GetTimeCreated", "FileSpec")]

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

Parameters

FileSpec: *the Filespec or 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.

Example Usage

Set Field[creationTime, External("TrFile-GetTimeCreated", "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 ScriptMaker add the following script step:

Set Field[creationTime, External("TrFile-GetTimeCreated", gFilePath)]

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

TrFile-GetTimeModified

Syntax Set Field [result, External("TrFile-GetTimeModified", "FileSpec")]

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

Parameters

FileSpec: *the Filespec or 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.

Example Usage

Set Field[modificationTime, External("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 ScriptMaker add the following script step:

Set Field[modificationTime, External("TrFile-GetTimeModified", gFilePath)]

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

TrFile-Launch

Syntax Set Field [result, External("TrFile-Launch", "FileSpec")]

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

fileSpec: *the Filespec or path to the file to be launched.*

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).

Example Usage

Set Field[result, External("TrFile-Launch", "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 ScriptMaker add the following script step:

Set Field[gErrorCode, External("TrFile-Launch", gFilePath)]

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

TrFile-ListDisks

Syntax Set Field [result, External("TrFile-ListDisks", "")]

Lists the names of all currently available disks.

Parameters

none, leave blank

Returned result

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

Example Usage

Set Field[result, External("TrFile-ListDisks", "")]

This will return a list of names of all the disks that are currently mounted, for example:

KES¶
SPOCK¶
Fotos tm 990926¶
Fotos tm 991115

TrFile-ListFolder

Syntax Set Field [result, External("TrFile-ListFolder", "switches | FileSpec")]

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.*
FileSpec: *the FileSpec or 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*

Returned result

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

Special Considerations

Note that each item, including the last, is followed by a return.

Example Usage

Set Field [result, External("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, and also the & is just there for readability. This might be returned:

Test.fp3¶
Desktop Folder¶
Program files¶
etc...

On Windows an example usage is:
Set Field [result, External("TrFile-ListFolder"; "files & showshortcuts|C:\Data\")]

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:

TrFile-ListFolder

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 ScriptMaker add the following scriptstep:

```
Set Field[gFolderList, External("TrFile-ListFolder", "files & folders |" & gFolderPath )]
```

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

TrFile-MetaData

Syntax Set Field [result, External("TrFile-MetaData", "switch|FileSpec")]

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.*
FileSpec: *the FileSpec or path to the file*

switches can be one or more of the following:

-getIPTC	*get the IPTC metadata (Photoshop's File Info)*
-getExif	*get the Exif metadata (A standard for Digital camera's)*
-getJPEGComment	*get the JPEG comment metadata of a JPEG file*

Returned result

metadatathe requested metadata, if available.

Special Considerations

Note that the data is returned in a raw form. At this moment we are still working on this. email us if you want more info on this.

Example Usage

External("TrFile-MetaData", "-getIPTC |" & gFilePath)

TrFile-MoveFile

Syntax Set Field [result, External("TrFile-MoveFile", "source FileSpec | destination FileSpec")]

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

Parameters

source FileSpec:	*the path to the file to move.*
destination FileSpec:	*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
$$-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-Get FileSpec Dialog" and "TrFile-Save FileSpec Dialog" to get a FileSpec for a file.

This function only works for source and destination on the same disk.

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 ScriptMaker add the following scriptstep:

```
Set Field[gErrorCode, External("TrFile-MoveFile", gSourceFilePath & "|" & gDestFilePath )]
```

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

TrFile-MoveFile

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. In ScriptMaker create the following script:

```
Set Field[gErrorCode, External("TrFile-MoveFile", "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-ReferenceToClip

Syntax Set Field [result, External("TrFile-ReferenceToClip", "Switch | FileSpec | PathToFMFile")]

With this function you can import an picture and store only a reference to it in FileMaker Pro. You can import the following graphic types: GIF, JPEG, PICT, EPS.

Parameters

Switch:	*reserved for future use, leave empty or set to "-unused"*
FileSpec:	*the Filespec or path to the picture file*
PathToFMFile:	*the path to the current FileMaker file (optional, but see below)*

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 picture file reference is put on the clipboard
$$-43	fnfErr	Picture file not found
$$-50	paramErr	Parameter error, check if the supplied parameters are correct

Other errors may be returned.

Special Considerations

This functions changes the clipboard. You can save and restore the clipboard with the Troi ClipSave plug-in.

The 3rd parameter (pathToFMPFile) is optional, but recommended. With this path to the current FileMaker file the plug-in can also calculate a relative path to the image file, making finding the picture reference more robust. Use the function TrFile-GetPathTo to get the path of the current FileMaker file.

Example Usage

Set Field[gErrorCode, External("TrFile-ReferenceToClip", "-unused |C:\Images\Pict1.JPG")]

This command will put a reference of the JPEG picture Pict1 on the clipboard. This can be pasted into a container field.

Example 2

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

containerField	container
gErrorCode	Global, number
gSelectedFile	Global, text
gPathToCurrentFMFile	Global, text

gPathToCurrentFMFile should contain the path to the current file (The best way to set this is during a startup script). gSelectedFile should contain the path to an existing picture file. In ScriptMaker add the following scriptsteps:

Comment [First get the imageRef on the clipboard...]
Set Field [gErrorCode, External("TrFile-ReferenceToClip",

TrFile-ReferenceToClip

```
                                        "-unused |"  & gSelectedFile & "|" & gPathToCurrentFMFile)]
If [gErrorCode = 0]
  Comment [Paste the reference in the container field]
  Paste [Select, containerField]
End If
```

TrFile-Save FileSpec Dialog

Syntax Set Field [result, External("TrFile-Save FileSpec Dialog", "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 chozen file. Use this FileSpec for other File plug-in functions.

Parameters

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 FileSpec or 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.

Example Usage

```
Set Field[myFileName, External("TrFile-Save FileSpec Dialog",
                        "Please choose a file to create|myFile.txt|" & gInititialDir)]
If [Left(myFileName, 2) <> "$$"]
  Set Field[gErrorCode, External("TrFile-CreateFile", myFileName )]
Endif
```

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

TrFile-Search

Syntax Set Field [result, External("TrFile-Search", "switches | Volume | SearchName")]

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

Parameters

switches:	*these determine the items that are listed.*
volume:	*the name of the volume on 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*

Returned result

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

Special Considerations

•At the moment you can only search a complete volume.•

Example Usage

```
Set Field [ result, External("TrFile-Search",  "files &folders |KES: |readme")]
```

This will return a list of all files and folders on the hard disk "Mac HD" 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:

```
KES:Foto, film, geluid:Geluiden:Alert!!!:Readme¶
KES:Programma's:Beeld:GIFConverter 2.3.7 ƒ:README-registration¶
KES:Programma's:Beeld:•MOVIE:OBJTOOL:README, Make QTVR Object¶
etc...
```

On Windows an example use is:

```
Set Field [ result, External("TrFile-Search",  "files &folders |C:|readme")]
```

TrFile-SelectFolderDialog

Syntax Set Field [result, External("TrFile-SelectFolderDialog", "switches | prompt | path to start")]

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

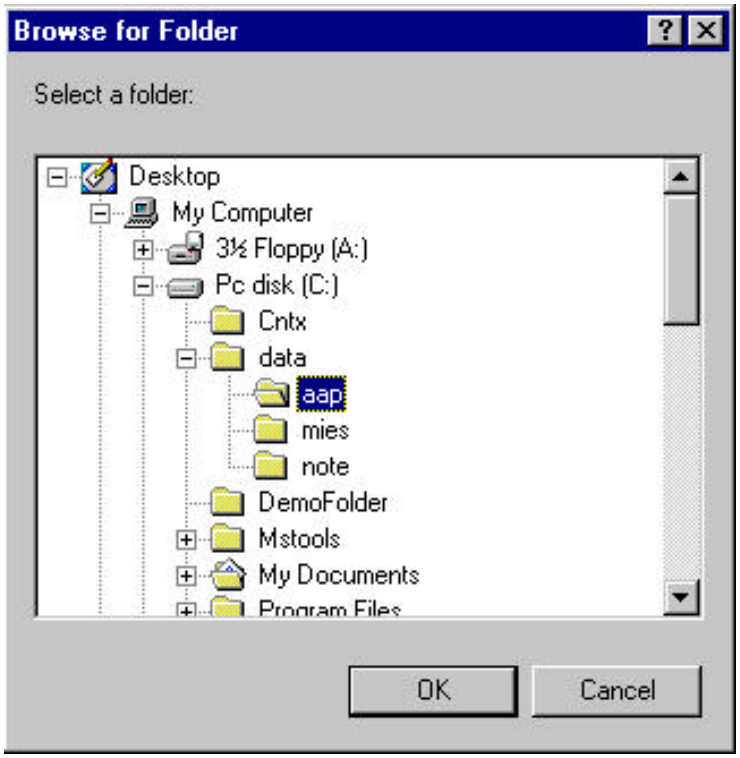
Parameters

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.*
path to start:	*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, External("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\".

TrFile-SelectFolderDialog

Example 2

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 ScriptMaker add the following script step:

```
Set Field[gFolderPath, External("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 Set Field [result, External("TrFile-SetContents", "text")]

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

Parameters

text: the characters that you want to write in the 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-Save FileSpec Dialog" to get a FileSpec for the file,
“TrFile-SetDefaultFileSpec” to indicate the file to affect.

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. In ScriptMaker create the following script:

```
Set Field[gErrorCode, External("TrFile-SetDefaultFileSpec", "C:\Testtext.txt" )]
Set Field[gErrorCode, External("TrFile-SetContents", "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 ScriptMaker add the following scriptstep:

```
Set Field[gErrorCode, External("TrFile-SetDefaultFileSpec", gFilePath )]
Set Field[gErrorCode, External("TrFile-SetContents",   textField)]
```

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

TrFile-SetDefaultCreator

Syntax Set Field [result, External("TrFile-SetDefaultCreator", "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

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 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.
Mac only. Ignored on Windows.

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 ScriptMaker add the following scriptstep:

```
Set Field[gErrorCode, External("TrFile-SetDefaultFileSpec", gFilePath )]
Set Field[gErrorCode, External("TrFile-SetDefaultCreator",   "ttxt")]
Set Field[gErrorCode, External("TrFile-SetDefaultFileType",  "TEXT")]
Set Field[gErrorCode, External("TrFile-SetContents",   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 Set Field [result, External("TrFile-SetDefaultFileSpec", "FileSpec")]

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

Parameters

FileSpec: *the Filespec or path to the file.*

Returned result

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

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 ScriptMaker add the following scriptsteps:

```
Set Field[gErrorCode, External("TrFile-SetDefaultFileSpec", gFilePath )]
Set Field[gErrorCode, External("TrFile-SetDefaultCreator",   "ttxt")]
Set Field[gErrorCode, External("TrFile-SetDefaultFileType",  "TEXT")]
Set Field[gErrorCode, External("TrFile-SetContents",   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 Set Field [result, External("TrFile-SetDefaultType", "[FileType1FileType2FileType3FileType4]")]

Sets the FileType to be displayed when browsing GetFile dialogs. Also sets the default file type when creating new files.

Parameters

If no parameter is specified, the "TrFile-Get FileSpec Dialog" in the new function will display all files.

Individual file type must contain 4 characters and is case sensitive. Add more file types without separators.

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.
Mac only. Ignored on Windows.

Example Usage

Set Field[gButtonNr, External("TrFile-SetDefaultType", "TEXTttro") to display SimpleText and Teach Text read-only files.

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 ScriptMaker add the following scriptsteps:

```
Set Field[gErrorCode, External("TrFile-SetDefaultFileSpec", gFilePath )]
Set Field[gErrorCode, External("TrFile-SetDefaultCreator",   "ttxt")]
Set Field[gErrorCode, External("TrFile-SetDefaultType",  "TEXT")]
Set Field[gErrorCode, External("TrFile-SetContents",   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-SetFileAttribute

Syntax Set Field [result, External("TrFile-SetFileAttribute", "switches|FileSpec")]

Sets an attribute of the file specified by the FileSpec.

Parameters

switches: *these determine the attribute that is set.*
FileSpec: *the FileSpec or path to the file.*

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)*

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".

Example Usage

```
Set Field[result, External("TrFile-SetFileAttribute", "-lock|C:\test.doc")]
```

The file will be locked (made Read-Only) and the result will be 0 (provided the file exists).

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

gFileSpec 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[gErrorCode, External("TrFile-SetFileAttribute",
                        If (gLockIt = 1, "-lock","-unlock") & "|" & gFileSpec)]
```

TrFile-SetFileAttribute

This will make the file locked if gLockIt is 1 and unlocked otherwise.

TrFile-Substitute

Syntax Set Field [result, External("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 Filespec or path to the file which is the source for the substitution.*
destinationFile: *the Filespec or 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.*

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 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.

Example Usage

```
Set Field[result, External("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.

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"

TrFile-Substitute

(Windows) or "Mac HD:Output:Log 2000_01" (Mac). In ScriptMaker add the following scriptstep:

```
Set Field[gErrorCode, External("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-ThumbnailToClip

Syntax Set Field [result, External("TrFile-ThumbnailToClip", "Switches | FileSpec")]

Creates a thumbnail (a small image of normally 80x80 pixels) from a graphics file and puts it on the clipboard.

Parameters

Switches:	*these determine the properties of the thumbnail that is returned.*
FileSpec:	*the Filespec or path to the file from which to create a 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 length or width is 80 pixels*
-Size=128	*maximum size of the length or width is 128 pixels*
-Size=256	*maximum size of the length or width is 256 pixels, more like a preview =)*
-BitDepth=8	*use 8 bits (=256 colors) instead of 16 bits (=65536 colors)*

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 thumbnail was created
$$-2020	movieToolboxUninitialized	QuickTime not available
$$-2003	cantFindHandler	No thumbnail could be created

Other errors may be returned.

Special Considerations

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.

NOTE1: When you want to paste from the clipboard the container field needs to be on the current layout.
NOTE2: When unsuccessful the clipboard can still be emptied by the plug-in.

•You need to have QuickTime installed in order to view the pictures in the database!•

NOTE: version 2.1 of the plug-in adds this function on the Windows platform

Example Usage

```
Go To Layout[container Layout]
Set Field [ result, External("TrFile-ThumbnailToClip", "-unused |Mac HD:photo.jpeg" )]
Paste[container field]
```

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

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

TrFile-ThumbnailToClip

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 ScriptMaker add the following script steps:

```
Go To Layout[ container Layout]
Go to Record[first]
Loop
   Set Field[gErrorCode, External("TrFile-ThumbnailToClip", "-unused|" & filePath)]
   Paste[thumbnailField]
   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-UnmountDisk

Syntax Set Field [result, External("TrFile-UnmountDisk", "diskname")]

Unmounts and ejects a removable disk. Mac OS only.

Parameters

diskname *the name of the disk to be 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	File is busy,
$$-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.

Example Usage

Set Field[result, External("TrFile-UnmountDisk", "Mac HD:")]

This will eject the hard disk "Mac HD", provided it is not the startup disk.

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 ScriptMaker add the following script step:

Set Field[gErrorCode, External("TrFile-UnmountDisk", gDiskName)]

This will eject the disk specified by gDiskName.

TrFile-Version

Syntax Set Field [result, External("TrFile-Version", "")]

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

No parameters, leave empty for future use.

Returned result

The name and version number of the loaded plug-in.
The function returns "" if this plug-in is not loaded.

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 = External("TrFile-Version, "")
```

This will evaluate to "Troi File Plug-in <version number>". This currently returns "Troi File Plug-in 2.0b1".