Welcome to *ExportFM*

Introduction

The **ExportFM** plug-in allows you to manipulate images within a FileMaker Pro database and export those images – as well as sounds, movies, and text – as documents in their original file formats. Types supported are GIF, JPEG, PICT, BMP, sound, and text. **ExportFM** handles any image or media type when stored by reference, as well as most of the types that FileMaker can store directly in container fields.

ExportFM8 works with FileMaker Pro 8 and 7 on Mac OS X and Windows.

*With **ExportFM**, you can…*

- Create a fast, dynamic web site from your database with the click of a button! The web site is all in HTML – no special web server required, and since no database server is required it's hostable on any web server and pages can be indexed by search engines and bookmarked by users!

- Crop, resize, and rotate images stored or referenced in a container field.

- Export FileMaker container field contents (graphics, sounds, movies) as files in their native file format (JPEG, GIF, PICT, BMP, snd, wav, etc.).

- **ExportFM** can convert GIF, PICT, or BMP image types to JPEG with user specified parameters for size, resolution, bit depth, and quality – enabling the creation of thumbnails suitable for web sites, or simply for the purpose of image manipulation.

- Export Text from text or calculation fields to a text file (including as an HTML file), setting the name and location based on field values or by script.

Installation

Installation

Place the ExportFM plug-in into the FileMaker "Extensions" folder inside your FileMaker Pro folder.

Make sure ExportFM is selected in the Plug-Ins section of the Application Preferences (found in the Edit Menu in Windows and in the FileMaker Pro menu in Mac OS X).

Demo Mode:

While ExportFM is unregistered it operates in demo mode. It will work up to 30 minutes or for 20 exports. Quit and restart FileMaker to get another demo session.

Registration:

Before you can register your copy of ExportFM, you must first purchase a user license. To order your user license, please visit the New Millennium Communications, Inc website at <http://www.newmillennium.com> or contact us by email at plug-ins@nmci.com. You can also order your license by selecting one of the order buttons in the ExportFM Demo file's Register layout.

Once you have your registration license, you can register ExportFM in the Demo files by entering your Licensee Name and Registration Codes in the Register layout and clicking the Register button.

Important: When incorporating ExportFM into your own FileMaker solutions, you must register ExportFM EVERY TIME you launch FileMaker Pro. A simple way to do this is to include a registration script step in your solution's On Open script. If you have multiple files in your database, you should either:

1) Include the registration step in the On Open script for each database, or
2) Require that the primary file be launched to ensure that its On Open script will register ExportFM.

An alternate approach is to simply re-register ExportFM at the beginning of any script that uses an ExportFM function.

For more information on registering ExportFM in your FileMaker solutions, please see the details on the Export-Register function.

QuickTime Note:

A few of the ExportFM functions require QuickTime 4.0 or greater. If you do not already have this installed on your computer, it can be downloaded for free at <http://www.apple.com/quicktime/download/>.

New Millennium
COMMUNICATIONS

New Millennium Communications
1332 Pearl Street
Boulder, CO 80302 USA
303-444-1476

plug-ins@nmci.com
www.newmillennium.com

New Millennium Communications is a software development company located in Boulder, Colorado. We specialize in making sophisticated tools for FileMaker Pro developers.

Table of Contents

ExportFM Core Functions

Export-Export

External("Export-Export", "FileName|FileType|Creator")

Export-Export exports an image, text, sound, or movie file and places it in the current destination folder. Any existing document with the same name will be replaced.

Putting Images on the Clipboard

ExportFM's Export command exports images, text, sounds, and movies from the clipboard. To place an object on the computer's clipboard, select it and copy it. This can be done manually or by script. To copy an object in a script, use FileMaker's 'Copy' script step and select the field holding the object:

Copy [Select; Table::Graphic Container Field]
or
Copy [Select; Table::Text Field]
etc.

File Types

Graphic Image Types

With images, there may be more than one image type on the clipboard after copying a container field. You must, therefore, specify the image format you want to use in the FileType parameter. If you are uncertain which image types are on the clipboard, you can use ExportFM's GetTypes command (see Export-GetTypes for more details).

Sound Types

All exported sound files must be specified as "snd" files, regardless of platform. When exporting a sound file on Windows, ExportFM will automatically convert the file to the "wav" format used by Windows.

Exporting Text

Exported text is exported in plain text format on both Windows and OS X. Text formatting is not maintained in the exported text.

Referenced Files

Pictures in a database may be stored by reference if "Store Only a Reference to the File" was checked at the time the picture was originally imported via the "Insert Picture" command in the File menu.

In this case, it is not the actual data that is saved in the container field, but a reference to the file's location. When the field is copied, an item of type "ref" is placed on the clipboard. Specify "ref" as the type. ExportFM will find the original file and copy it to the destination folder.

Movie files can only be stored by reference.

Creators (Macintosh only)

The creator code indicates which application should be launched when the document is opened (such as by double-clicking the file). If the application with the given creator code is not available, the Mac displays a list of all applications capable of opening a document of that type, and asks you to choose one.

To find out the creator code of a particular document or application, use a utility like ResEdit, Snitch, or Norton Disk Editor.

On OS X, the creator code is optional, but the parameter must still be filled in. If you use the generic code "****", a default application will be selected based on either the file type or the three letter filename extension (".txt", ".jpg", etc.).

On Windows, the creator is ignored.

20 Exports in Demo Mode:

When ExportFM is unregistered, it will only execute the Export command 20 times. After Export-Export is used for the twentieth time, ALL ExportFM commands will cease to function and will, instead, return the error "$999:Not Registered." This will occur even if you have been testing the plug-in for less than 30 minutes. If you wish to continue testing your unregistered version of ExportFM, you must restart FileMaker.

If ExportFM is registered, you can, of course, use the Export command without limit.

Errors

If Export is successful, it returns an empty string. When there is an error, you get an error message prefixed with the specific Mac OS or Windows error code. The error messages are often generic - many different file system errors report "Could not create/write export file".

$999: Not registered (ExportFM is not registered and the evaluation period has expired.)
$998: Error in parameter format
$xxxx: The requested type is not in the clipboard
$xxxx: Referenced file not found (Error resolving a reference.)
$xxxx: Error while copying referenced file
$xxxx: Could not create/write export file (Disk full, privilege error, and many others.)

Parameters:

FileName – The name of the document to be exported. (Windows: don't include the three letter extension in FileName. The correct extension will be appended automatically.) When in demo mode, the word "DEMO" will be appended to the filename.

FileType – The filetype code of the object to be exported.

gif = GIF image
jpg = JPEG image
pct = PICT image
bmp = Bit Mapped image
txt = Text
snd = Sound file
ref = Referenced Graphic or Movie

For any other filetypes, ExportFM will return an error.

Creator – The four character creator code for the exported file.
On OS X, the creator code is optional, but it still must be filled in. If you use the generic code "****", a default application will be selected based on either the file type or the three letter filename extension (".txt", ".jpg", etc.).
Windows chooses the file's application based only on the filename's three letter extension. The creator code parameter is ignored in the Windows version.

"ttxt" = TextEdit (Mac OS X)
"****" = Default application (Mac OS X)

The parameters must be separated by a pipe character "|".

Examples:

To export a JPEG image (stored in a field named "Graphic Container") on OS X--

```
Copy [Select; Table::Graphic Container]
Set Field [Table::Response Field; External("Export-Export"; "Image|jpg|****")]
```

To export a PICT image to open in TextEdit on Mac OS X--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-Export"; "Image|pct|ttxt")]
```

To export a JPEG image on Windows (with no creator code), naming the exported file using the name in a field called Filename--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-Export"; Filename & "|jpg|")]
```

On OS X, to export a GIF image to the desktop (see Export-SetDestinationFolder for more details)--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field [Table::Response Field; External("Export-Export"; "Smiley|GIF|****")]
```

On Windows, to convert a PICT image to a JPEG, and then export it to the desktop (see Export-ConvertImage for more details)--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-ConvertImage"; "pct||||")]
Set Field [Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field [Table::Response Field; External("Export-Export"; "SunnyDay|jpg|")]
```

To export text to the desktop on OS X--

```
Copy [Select; Table::Text Field]
Set Field [Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field [Table::Response Field; External("Export-Export"; "Document|txt|ttxt")]
```

To export text to the desktop on Windows--

```
Copy [Select; Table::Text Field]
Set Field [Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field [Table::Response Field; External("Export-Export"; "PlainText|txt|")]
```

On OS X, to export a sound file (SND format)--

```
Copy [Select; Table::Sound Container Field]
Set Field [Table::Response Field; External("Export-Export"; "Wahoo|snd|****")]
```

On Windows, to export a sound file (ExportFM automatically converts it to WAV format)-

```
Copy [Select; Table::Sound Container Field]
Set Field [Table::Response Field; External("Export-Export"; "Huzzah|snd|")]
```

To export a movie file stored by reference in your database as a QuickTime Player document, using the name in the Movie Name field--

```
Copy [Select; Table::Movie Container Field]
Set Field [Table::Response Field; External("Export-Export"; MovieName&"|ref|TV0D")]
```

Export-ConvertImage

External("Export-ConvertImage", "ImageType|Vert|Horiz|BitDepth|Quality|Resolution")

Lets you adjust the dimensions, color quality, compression quality and resolution of an image on the clipboard. The resulting image is always JPEG format. Extremely useful for making quick thumbnails or exporting stored PICT files as JPEGs for use on a website or as an email attachment.

Putting Images on the Clipboard

ConvertImage affects images on the clipboard. To place an image on the computer's clipboard, select it and copy it. This can be done manually or by script. To copy an image in a script, use FileMaker's 'Copy' script step and select the container field holding the image:

```
Copy [Select; Table::Graphic Container Field]
```

Image Types

Since there may be more than one image on the clipboard after copying a container field, you must specify the image type you want to use in the ImageType parameter. If you are uncertain which image types are on the clipboard, you can use ExportFM's GetTypes command (see Export-GetTypes for more details).

The next two parameters are numbers, the vertical and horizontal limits for the thumbnail. ConvertImage then fits the original image into the given rectangle, preserving the image's proportions. (Since ConvertImage does not distort the image's proportions, the thumbnail may only touch the edges of the rectangle in one dimension.) You can further specify color or gray-scale bit depth, image compression quality, and resolution.

The resulting image is placed on the clipboard. It can then be pasted into a container field or exported.

QuickTime

ConvertImage requires QuickTime 4.0 or greater. If you do not already have this installed on your computer, it can be downloaded for free at <http://www.apple.com/quicktime/download/>.

If QuickTime is not installed, ExportFM will return an error. You can use ExportFM's CheckQT command to verify that QuickTime 4.0 or greater is installed.

Sound and Text Files

Do not use ConvertImage on sound or text files. ExportFM will return an error.

White Circle in Demo Mode:

When you use ExportFM's ConvertImage command in demo mode (that is, when ExportFM is unregistered), a white circle will be placed over the image. This does not occur with a registered version of ExportFM.

Viewing the Results

After the Export-ConvertImage step, the image is still only in the invisible clipboard. To see the resulting image, you must paste it into a container field using the PasteToContainer function with a script step such as:

Set Field [Table::Container Field; PasteToContainer ("jpg")]

(See the PasteToContainer function.)

Or you can export the image file with a step like:

Set Field [Table::Response Field; External("Export-Export"; "Thumbnail.JPG|jpg|ttxt")]

(See the Export-Export function.)

Errors

Any errors are returned as text; an empty string indicates success.

If ConvertImage does not work on Windows 2000 or XP

The ConvertImage command can encounter problems with certain BMP image files on Windows 2000 and XP. If you are having problems with the ConvertImage command on Windows 2000 or XP, you may need to adjust the display settings for your monitor. In the Display control panel, go to the Settings tab and adjust the color resolution downward to 256 colors. Then try the ConvertImage command again to see if the problem is solved.

Parameters:

ImageType – The 3 letter type of the input image
- gif = GIF image
- jpg = JPEG image
- pct = PICT image
- bmp = Bit Map image
- ref = Referenced Image

Vert - The maximum vertical dimension (in pixels) of the image. Leave blank if you want to maintain the original image's dimensions. For thumbnail images, 80 pixels is a common maximum dimension.

Horiz – The maximum horizontal dimension (in pixels) of the image. Leave blank if you want to maintain the original image's dimensions. For thumbnail images, 80 pixels is a common maximum dimension.

BitDepth – The color or gray-scale bit depth for the resulting image. Smaller bit depths give more compact images that load and display faster and require less storage space, but they have fewer colors or gray-scale range. Leave blank if you want to maintain the original image's bit depth.

1 = Color – Bit Depth 1
2 = Color – Bit Depth 2
4 = Color – Bit Depth 4
8 = Color – Bit Depth 8
16 = Color – Bit Depth 16
24 = Color – Bit Depth 24
32 = Color – Bit Depth 32
33 = Gray-Scale – Bit Depth 1
34 = Gray-Scale – Bit Depth 2
36 = Gray-Scale – Bit Depth 4
40 = Gray-Scale – Bit Depth 8

Some image types do not allow all of these depth choices. ConvertImage will default to the next larger allowable depth.

Quality – The quality of the image compression. Higher compression levels yield smaller image files which load and display faster, but have lower image quality. Leave blank if you want to maintain the original image's compression quality.

0 = Minimum Quality, Maximum Compression
256 = Low Quality, High Compression
512 = Normal Quality, Medium Compression
768 = High Quality, Low Compression
1023 = Maximum Quality, Minimum Compression
1024 = Lossless Quality

Some image types do not implement all of these compression values. ConvertImage will default to the next larger allowable quality.

Resolution – The resolution of the image in dots per inch. The standard for screen display is 72. Lower resolution yields a smaller image file that is coarser visually. Leave blank if you want to maintain the original image's resolution.

The parameters must be separated by a pipe character "|".

Examples:

To convert a PICT image (stored in a field named "Graphic Container") to a JPEG, without affecting the images size--

```
Copy [Select; Table::Graphic Container]
Set Field [Table::Response Field; External("Export-ConvertImage";"pct|||||")]
```

To convert a GIF image to a thumbnail-sized JPEG, setting the maximum dimensions to 80 x 80 pixels--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-ConvertImage";"gif|80|80|||")]
```

To convert an image stored by reference into a JPEG, shrinking it to thumbnail size, and then store it in a separate container field (see the PasteToContainer function for more info) --

```
Copy [Select; Table::Graphic Container Field 1]
Set Field [Table::Response Field; External("Export-ConvertImage";"ref|80|80|||")]
Set Field [Table::Graphic Container Field 2; PasteToContainer ("jpg") ]
```

To convert a JPEG image to a 50 x 50 gray-scale thumbnail--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-ConvertImage";"jpg|50|50|36||")]
```

To convert a high-resolution JPEG image to 72dpi--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-ConvertImage";"jpg||||72")]
```

To convert a PICT image to a thumbnail-sized JPEG, highly compressing the image--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-ConvertImage";"pct|80|80||256|")]
```

Since the ConvertImage command requires a recent version of QuickTime to be installed and active, it is a good idea to first check to make sure QuickTime is available. (See Export-CheckQT for more details.)--

```
If [External("Export-CheckQT";"") <>1]
  Show Message ["You must have QuickTime 4.0 or greater installed to run this script."]
  Halt Script
End If
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-ConvertImage";"pct|80|80||256|")]
```

To convert a PICT image to a 72dpi thumbnail JPEG, and then export it to the desktop (see Export-Export and Export-SetDestinationFolder for more details)--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-ConvertImage";"pct|80|80|||72")]
Set Field [Table::Response Field; External("Export-SetDestinationFolder";".D")]
Set Field [Table::Response Field; External("Export-Export";"Thumbnail.JPG|jpg|ttxt")]
```

To convert a PICT image to a 72dpi thumbnail JPEG, and then paste it back into the original container field (see PasteToContainer for more details)--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-ConvertImage";"pct|80|80|||72")]
Set Field [Table::Response Field; External("Export-SetDestinationFolder";".D")]
Set Field [Table::GraphicContainer Field; PasteToContainer ("jpg")]
```

Export-CropImage

External ("Export-CropImage", "Top|Left|Bottom|Right")

Lets you crop the image in the clipboard during the next Export-ConvertImage step.

The Export-CropImage command sets the dimensions for a rectangle crop of the image carried on the clipboard. The image is not actually cropped until the next Export-ConvertImage command is called.

Original Image Dimensions

The crop rectangle dimensions are based on the original dimensions of the source image. On the original image, top left is 0,0. The original image's bottom is the number of pixels in its height. It's right dimension is the number of pixels in its width.

For example, if the original image was 75 pixels wide by 125 pixels high, its dimensions would be

```
0|0|125|75
```

Use Export-GetImageInfo

You can use the Export-GetImageInfo to get the original image's dimensions.

Export-CropImage Dimensions

If you want to crop 10 pixels from all four edges of the image, you would need to set your parameters to--

```
10|10|115|65
```

The coordinates you specify in the Export-CropImage command are automatically limited to the original image's dimensions. If you attempt to define dimensions greater than the source image's width and height, the image outer edge will remain unchanged in that dimension.

Using with Copy and Export-ConvertImage

The Export-CropImage command only takes effect when the next Export-ConvertImage command occurs. And the image must already be in place on the clipboard. The three necessary script steps should occur in this order:

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-CropImage"; "10|10|115|65")]
Set Field [Table::Response Field; External("Export-ConvertImage"; "gif|||")]
```

Viewing the Results

After the Export-ConvertImage step, the image is still only in the invisible clipboard. To see the resulting image, you must paste it into a container field using the PasteToContainer function with a script step such as:

```
Set Field [Table::Container Field; PasteToContainer ("jpg")]
```

(See the PasteToContainer function.)

Or you can export the image file with a step like:

```
Set Field [Table::Response Field; External("Export-Export"; "Thumbnail.JPG|jpg|ttxt")]
```

(See the Export-Export function.)

The crop rectangle is reset after each Export-ConvertImage command.

Note: Because the Export-ConvertImage command is used, the resulting image is always in JPEG format.

Parameters:

Top – The crop rectangle's top boundary. This parameter should not be less than zero. (The source image's original top boundary is 0.)

Left – The crop rectangle's left boundary. This parameter should not be less than zero. (The source image's original left boundary is 0.)

Bottom – The crop rectangle's bottom boundary. This parameter should not be greater than the height of the source image. (The source image's original bottom boundary is the number of pixels in its height.)

Right – The crop rectangle's right boundary. This parameter should not be greater than the width of the source image. (The source image's original right boundary is the number of pixels in its width.)

The parameters must be separated by a pipe character "|".

Examples:

To create a JPEG that crops 10 pixels from each edge of a source GIF image that was originally 125 pixels high and 75 pixels wide, and then paste the results into the original field--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-CropImage";"10|10|115|65")]
Set Field [Table::Response Field; External("Export-ConvertImage";"gif|||")]
Set Field [Table::GraphicContainer Field; PasteToContainer ("jpg")]
```

To split a 400 x 400 JPEG into two separate fields, dividing the image into left and right halves--

```
Copy [Select; Table::Source Container Field]
Set Field [Table::Response Field; External("Export-CropImage";"0|0|400|200")]
Set Field [Table::Response Field; External("Export-ConvertImage";"jpg|||")]
Set Field [Table::Left Container Field; PasteToContainer ("jpg")]
Copy [Select; Table::Source Container Field]
Set Field [Table::Response Field; External("Export-CropImage";"0|200|400|400")]
Set Field [Table::Response Field; External("Export-ConvertImage";"jpg|||")]
Set Field [Table::Right Container Field; PasteToContainer ("jpg")]
```

Export-RotateImage

External("Export-RotateImage", "RotateDeg")

Sets the degrees of image rotation for the next Export-ConvertImage command.

The Export-RotateImage command specifies the number of degrees to rotate the image in a clockwise direction during the next Export-ConvertImage command. The image is rotated around the center of the original image.

Using with Copy and Export-ConvertImage

As with Export-CropImage, the Export-RotateImage command only takes effect with the next Export-ConvertImage command. The source image must already be in place on the clipboard. The three necessary script steps should occur in this order:

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-RotateImage","180")]
Set Field [Table::Response Field; External("Export-ConvertImage","gif|||")]
```

Viewing the Results

After the Export-ConvertImage step, the image is still only in the invisible clipboard. To see the resulting image, you must paste it into a container field using the PasteToContainer function with a script step such as:

```
Set Field [Table::Container Field; PasteToContainer ("jpg")]
```

(See the PasteToContainer function.)

Or you can export the image file with a step like:

```
Set Field [Table::Response Field; External("Export-Export"; "Thumbnail.JPG|jpg|ttxt")]
```

(See the Export-Export function.)

Because the Export-ConvertImage command is used, the resulting image is always in JPEG format.

Rotation is reset after each Export-ConvertImage command.

Rotating Other Than 90 Degrees

Export-RotateImage is designed to work with rotations in multiples of 90 degrees of arc (90, 180, 270). You can enter other rotation degree amounts, but filler space may be added beyond the edges of the image.

This added edge space can also affect the apparent size and position of the image if it is subsequently pasted into a container field -- depending on the field's graphic format settings. Simply adjusting the container field's graphic settings (Format>Graphics...) to "Crop" and "Center" will center the image and return it to its expected size.

Test thoroughly before rotating your images any amount other than multiples of 90 degrees to make sure that the results are acceptable.

Rotation and Image Boundaries

On OS X, a 90 degree rotation is rotates around a center point, no matter what graphic format alignment is set for the container field. If you are rotating something other than 90 degrees and the alignment is NOT set to center, then you get the odd affect of the image rotating off center and out of the container field's boundaries. However, setting the alignment back to center for the field will return the complete image back to center.

The behavior is different on Windows. A 90 degree rotation is not perfectly centered, even if the container field's graphic format is set to center the image. And any part of the image that moves outside of the container field's boundaries actually gets cut off

and is not restored by adjusting field alignment (or rotating the image again to bring the hidden part of the image back into the frame).

Image Quality on Windows

On Windows, if image quality seems to degrade, you may need to adjust the display settings to a higher color quality. You may also need to clarify clipping behavior of the container field displaying the image.

Parameter:

RotateDeg – The number of degrees to rotate the image in a clockwise direction.

Examples:

To create a JPEG that turns the source GIF image upside down--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-RotateImage"; "180")]
Set Field [Table::Response Field; External("Export-ConvertImage"; "gif|||")]
```

To rotate a JPEG image 90 degrees clockwise, and then see the results in the original Graphic Container Field--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-RotateImage"; "90")]
Set Field [Table::Response Field; External("Export-ConvertImage"; "jpg|||")]
Set Field [Table::GraphicContainer Field; PasteToContainer ("jpg")]
```

To rotate a JPEG image 90 degrees counter-clockwise, and then export it to the desktop--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-RotateImage"; "270")]
Set Field [Table::Response Field; External("Export-ConvertImage"; "jpg|||")]
Set Field [Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field [Table::Response Field; External("Export-Export"; "Rotated Image|jpg|TV0D")]
```

PasteToContainer

PasteToContainer (“ImageType”)

Allows you paste images from the clipboard to a container field.

In FileMaker 7 and 8, once an image has been modified on the clipboard with a function like Export-ConvertImage, you cannot subsequently paste the modified image to a

container field with a Paste script step. To work around this limitation, ExportFM uses the PasteToContainer function.

The simplest way to use the PasteToContainer function is to place it in a Set Field script step that specifies the desired container field:

```
Set Field [Table::Container Field; PasteToContainer ("jpg")]
```

You must specify the image type with the PasteToContainer function because more the image on the clipboard may be held in more than one format. You can, for example have the same image stored in both PCT and JPG formats.

If you are unsure which image types are available for a particular image on the clipboard, you can use the Export-GetTypes function.

Parameter:

ImageType – The 3 letter type of the image you want to paste from the clipboard to the container field

gif = GIF image
jpg = JPEG image
pct = PICT image
bmp = Bit Map image
ref = Referenced Image

Example:

To copy a GIF image to the clipboard, change it to a JPEG with Export-ConvertImage, and then paste it back into a new container field, your script steps might look like this—

```
Copy [Select; Table::Container GIF]
Set Field [Table::Response Field; External ("Export-ConvertImage"; "gif|||||")]
Set Field [Table::Container JPG; PasteToContainer ("jpg")]
```

File Control Functions

Export-CreateShortcut

External ("Export-CreateShortcut", "SourceFilename|DestFilename")

Creates a shortcut in the destination folder that points to the specified file in the source folder.

Export-CreateShortcut generates an alias file (on Mac) or shortcut file (on Windows) that points to the original file.

Source Folder and Destination Folder

Before calling the Export-CreateShortcut function, you must first define the source and destination folders. The Export-SetSourceFolder function specifies the actual location of the original file. Export-SetDestinationFolder defines where you want the resulting shortcut file to be placed.

As a simple example, let's say the original file is located on the Desktop and you want to place the resulting shortcut/alias in the FileMaker Application Folder. Before calling the Export-CreateShortcut function, you will first need the following steps:

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; ".D")]
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".A")]
```

This sets the source folder to be the Desktop and the destination folder to be the Application Folder. (See Export-SetDestinationFolder for a full description of using destination and source parameters.)

Creating the Shortcut

Once you have defined the source and destination folders, you can then create the shortcut file. You need to know the file name of the original file located in the source folder. The resulting shortcut file can be named any valid file name.

Building on our previous example, suppose the original file located on the Desktop is named "Export.tab" and we want the resulting shortcut that will appear in the Application Folder to be named "Export File Shortcut", the full script would be:

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; ".D")]
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".A")]
Set Field[Table::Response Field; External("Export-CreateShortcut"; "Export.tab|Export
    File Shortcut")]
```

As a variation on this example, suppose the original file name is calculated in a field called File Name Calc, then the final step would become:

```
Set Field[Table::Response Field; External("Export-CreateShortcut"; Table::File Name Calc
    & "|Export File Shortcut")]
```

(For more examples, see the Examples section of the Export-CreateShortcut documentation.)

Important Note:

The technique mentioned in the documentation for ExportFM 2.x (for use with FileMaker Pro 5/6) describing how to create a shortcut in order to attach variable images and files to emails does not work in Filemaker 7 or 8.

Errors

If the source file is not found in the source folder, ExportFM will return an error.

Also, if a file with the Destination FileName already exists in the destination folder, an error will be returned.

All error responses begin with a dollar sign "$", so you can test for errors using an If statement such as:

```
If [Left (Response Field, 1) = "$"]
    Beep
    Show Message ["An error has occurred."]
    Halt Script
End If
```

Parameters:

Source FileName – The name of the original file located in the source folder. This is the file that the new shortcut will point back to.

Destination FileName – The name to be given to the resulting shortcut file as it is created in the destination folder.

Examples:

To create a shortcut on the desktop named "Animals" that points to a source file named "Wild" in the FileMaker Application folder--

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; ".A")]
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field[Table::Response Field; External("Export-CreateShortcut"; "Wild|Animals")]
```

(See details on the Export-SetDestinationFolder and Export-SetSourceFolder functions to understand how to set locations.)

To create a shortcut on the desktop named "Click Me" that points to a source file in the Windows folder (Windows OS only), with the original file named stored in the File Name field--

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; ".W")]
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field[Table::Response Field; External("Export-CreateShortcut"; Table::File Name&
    "|Click Me")]
```

To create a shortcut (named simply "Shortcut") for a file selected by the user--

```
Set Field[Table::File Name; External("Export-SetSourceFolder"; ".V:Select a file")]
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".U:Select the
    location for the new alias")]
Set Field[Table::Response Field; External("Export-CreateShortcut"; Table::File Name&
    "|Shortcut")]
```

Export-MoveFile

External ("Export-MoveFile", "FileName")

Moves the specified file from the source folder to the destination folder. In conjunction with Export-RenameFile, this function gives you the ability to control exported files' names and locations. You can also use it to manipulate the location of image files that you have exported from container fields and to reorganize your files into different folders.

As the function's name suggests, Export-MoveFile gives you the ability to move a file from one location to another. You must specify a file name in the Export-MoveFile parameter. If a file with that name is found in the source folder (defined with the Export-SetSourceFolder function), the file will be moved to the destination folder (defined with the Export-SetDestinationFolder function).

Dynamic Exports

Used together with the Export-RenameFile function, Export-MoveFile becomes a powerful function that allows you to dynamically control any standard FileMaker Pro script step that generates an output file, such as Export Records.

These FileMaker Pro script steps normally require you to generate a file with a fixed name, placed in an unchanging location. Export-MoveFile gives you the ability to move the resulting file to any location, and Export-RenameFile allows you to change the file's name from the generic output name to any file name you choose, based on a field calculation or user-entered value, adding a date stamp, etc.

Using Export-MoveFile and Export-RenameFile together frees you from FileMaker's requirement that your exported file must be placed in a fixed location with a non-changing file name.

Source Folder and Destination Folder

Before calling the Export-MoveFile function, you must first define the source and destination folders. Use the Export-SetSourceFolder function to specify the original location of the file to be moved. Then use Export-SetDestinationFolder to define where you want the file to be moved to.

For example, if the file's original location is in FileMaker's Application Folder and you want to move it to the Desktop, you will first need the following steps:

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; ".A")]
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
```

This sets the source folder to be the Application Folder and the destination folder to be the Desktop. (See Export-SetDestinationFolder for a full description of using destination and source parameters.)

Using Export-MoveFile

Once you have defined the source and destination folders, you can then call the Export-MoveFile function. You need to know the name of the file located in the source folder.

Building on our previous example, suppose the file located in the Application Folder is named "Export.tab", the full script would be:

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; ".A")]
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field[Table::Response Field; External("Export-MoveFile"; "Export.tab")]
```

Using Move & Rename to Perform a Controlled Export

To do a controlled Export with ExportFM, it is **essential** to understand two issues: the rules that FileMaker follows regarding it's default location for placing exported files and the necessity of "breaking the path" after defining a script step -- both of which are documented below and in the in-depth examples.

As a simple Export example, suppose you want to export a file and have it end up on the desktop with the file name calculated in the File Name Calc field. Your script would look like this:

```
Export Records [Restore, No Dialog; "Export File"]
Set Field[Table::Response Field; External("Export-SetSourceFolder"; ".A")]
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
```

```
Set Field[Table::Response Field; External("Export-MoveFile"; "Export File")]
Set Field[Table::Response Field; External("Export-RenameFile"; "Export File|" & Table::File
    Name Calc)]
```

In the Export Records script step, mark the 'Restore Export Order' checkbox to save the export field order, and mark the 'Perform without Dialog' checkbox so the export dialog won't appear for the user when the script is run. The Export Records step stores the location where the file will be exported to (in this case, the Application Folder).

Why not export directly to the intended location in the first place?
The reason is that the file path stored in the Export Records script step will "break" if you move your files to a different machine. When that happens, the export location will revert to a "default location" and it will need to be moved. (See the Default Location explanation below.)

Export Order and Export File Type
If you want your script to allow for more than one export order (the order of the fields being exported) or export file format (tab-separated text, HTML Table, etc.), simply create several sub-scripts, each with its own Export Records script step. Define each Export Records step in each of the sub-scripts to export with a different export format. (The separate sub-scripts are necessary because FileMaker only allows you to specify one export order and export file type for each script.) You can then branch your script based on which button has been clicked in a message dialog, the value of a field calculation, etc., and call the appropriate export sub-script.

Exporting FileMaker Pro Format Files on OS X:
In FileMaker Pro for OS X, an error occurs when attempting to 'Save a Copy as' or 'Export Records' as a FileMaker Pro formatted file using the 'broken filepath' technique described here.

For this reason, DO NOT "break" the file path (as described below) when performing 'Export Records' in FMP format.

Using the 'fixed filepath' technique will only work for local files, however, not remotely hosted files. Exporting Records in other file formats (such as, tab-separated or comma-separated text, HTML tables, etc.) works with the 'broken filepath' technique described here.

Important: Default Locations and Breaking the File Path

One element that you must understand in using this "Move & Rename" technique is the issue of "default locations."

If you expect to use your solution files on several different machines or host them over a network, a file paths stored by FileMaker in the Export Records script step is likely to "break". That is, the file path stored in your script will no longer be valid for the new directories found on the new machines.

When the stored file path is no longer valid, FileMaker Pro will revert to a "default location" and send the output files there instead.

For that reason, it is important to construct your scripts assuming all activity will pass through this default location.

Breaking the File Path

When testing, it is important to first "break" the file path stored in the operational FileMaker Pro script step (Export Records). To break the stored file path:

- Create a temporary folder with any file name.
- In the Export Records script step, specify this temporary folder as the output location.
- Drag the temporary folder to the Trash or Recycle Bin. Empty the trash so the temporary folder no longer exists.

When the script is run, since FileMaker cannot locate the temporary file, it will then send the export file to the default location, instead.

Summary of Default Locations for Outgoing Actions (Export Records, Save a Copy as) using Move & Rename

- If the file is run as a **single user** or it is the **local host** (sharing may be turned on, but the file is not being run remotely over a network), the default location is the **Current File Folder.** That is, the default location is where the currently running FileMaker Pro file is located.

- If the file is run as a guest of a **remotely hosted file--**
  - On **Windows**, **Mac Classic**, and **OS X 10.2 and later**, the default location is the **Application Folder**.
  - On **Mac OS X pre-10.2**, the default location is the **Documents Folder**.

Be aware that **Save a Copy as** does not work on hosted files -- this is a FileMaker Pro issue; it has nothing to do with ExportFM.

For a detailed explanation of how to construct a sub-script to properly use the default location, see the in-depth example files. If you prefer, you can simply import one of the Set Default Location scripts from the in-depth example files and use it as a sub-script with your scripts in your own solution.

Errors:

If the file can't be found in the source folder, ExportFM will return an error.

An error is also returned if an identically named file already exists in the destination folder.

All error responses begin with a dollar sign "$", so you can test for errors using an If statement such as:

```
If [Left (Response Field, 1) = "$"]
    Beep
    Show Message ["An error has occurred."]
    Halt Script
End If
```

Export-RenameFile

External ("Export-RenameFile", "OldFileName|NewFileName")

Lets you rename any file. This can be very useful if you are exporting html text files on Windows -- you can change the ".txt" suffix to ".htm".

Once the location of the file has been specified using the Export-SetDestinationFolder command, you can then call the Export-RenameFile function. As long as the file is located in the destination folder and it is not locked or in use, ExportFM will change the file name for you.

Errors:

If the file can't be found (the old filename can't be found in ExportFM's destination folder), ExportFM will return:
$-43:Could not find specified file.

Locked files cannot be renamed. ExportFM will return this message:
$-45:Could not rename file (locked, duplicate name, no privileges)

Parameters:

OldFileName – This filename must exactly match the filename of the file to be changed (and it must be located in ExportFM's current destination folder).

NewFileName – This is what the filename will be changed to. When in demo mode, the word "DEMO" will be appended to the filename.

The parameters must be separated by a pipe character "|".

Example:

To change the name of a file (located on the desktop) from "TestFile" to "NewFile"--

```
Set Field [Table::Response Field; External ("Export-SetDestinationFolder"; ".D")]
Set Field [Table::Response Field; External ("Export-RenameFile"; "TestFile|NewFile")]
```

To change the name of a file from the name stored in Old Filename field to the name stored in New Filename field--

```
Set Field [Table::Response Field; External("Export-RenameFile"; Table::Old Filename & "|"
    & Table::New Filename)]
```

To change the name of a file and then test for errors --

```
Set Field [Table::Response Field; External("Export-RenameFile"; "TestFile|NewFile")]
If [Left(Table::Response Field, 5) = "$-43:"]
    Show Message ["The specified file could not be found."]
    Halt Script
End If
If [Left(Table::Response Field, 1) = "$"]
    Show Message ["The specified file exists, but could not be renamed."]
    Halt Script
End If
```

To change the name of a file, making sure the new filename isn't already taken by deleting any file with the new name--

```
Set Field [Table::Response Field; External("Export-DeleteFile"; "NewFile")]
Set Field [Table::Response Field; External("Export-RenameFile"; "TestFile|NewFile")]
If [Left(Table::Response Field, 1) = "$"]
    Show Message ["The specified file exists, but could not be renamed."]
    Halt Script
End If
```

To export the contents of a field named "HTML Text" as a text file with the name "webpage". Since, on Windows, the suffix ".txt" will automatically be added to the filename, we then want to rename the file to "webpage.htm"--

```
Copy [Select; "HTML Text"]
Set Field [Table::Response Field; External("Export-Export"; "webpage|txt|ttxt")]
Set Field [Table::Response Field; External("Export-RenameFile";
    "webpage.txt|webpage.htm")]
```

Export-SetDestinationFolder

External("Export-SetDestinationFolder", "FilePath")

Lets you tell ExportFM where to perform a subsequent function which involves a file's location. This is a powerful command with many options -- you can use a hard coded pathway, let the user specify a location, or specify a directory with a known location like the desktop or the FileMaker application folder.

You can specify where to perform the ExportFM functions that are dependent on a location:

Export-CreateShortcut
Export-MoveFile
Export-RenameFile
Export-NewFolder
Export-DeleteFile
Export-CopyFile
Export-ListDestinationFolder
Export-Open

Export-SetDestinationFolder is quite versatile and its FilePath parameter can take several forms:

1. Full Path

You can specify the full path to the destination folder. For example, you can use a full path like:

Macintosh HD:Desktop Folder:Exported Files:
or
C:\WINDOWS\Desktop\Exported Files\

Be aware that a full file path will become invalid if any folder in the path is moved or renamed. As a result, the other forms of specifying a path may be more reliable in many circumstances (see below).

Testing for Errors:
Because of the possible problems that can arise if the file path has changed, it is strongly recommended that you check for an error after setting the destination using the full path method. The Export-SetDestinationFolder command only returns a response when an error has occurred, so to check if an error occurred you can use:

If ["not IsEmpty (Table::Response Field)"]

The pathway format differs between Macintosh and Windows.
On Macintosh, the pathway begins with the disk name (for example, "Macintosh HD"). On Windows, the pathway must begin with the drive letter code (for example, "C:")

The separators used between different folders in the file path are different also. The Macintosh separator is a colon ":". The Windows separator is a backslash "\".

Also, when going up one level in the file path (see Relative Path below), you must use a double colon "::" on Macintosh and on Windows you must use a double period ".."

2. Special Codes

The following codes set the destination folder to specific folders:
".D" = The Desktop
".A" = The folder containing the active FileMaker application.

".S" = The OS System Folder
".P" = The Preferences Folder (Mac only)
".W" = The Windows Folder (Win only)
".T" = The Temporary Folder [see below for more details]
".F:" and the filename = Location of an open FileMaker database (Mac only) [see below for more details]
".O" = Documents Folder
".N" = Current Destination Folder (for the source folder)
".R" = Current Source Folder (for the destination folder)
".U:" and the prompt statement = User selected folder [see below for more details]
".V:" and the prompt statement = User selected file [see below for more details]

Temporary Folder
The Temporary Folder is a special folder located on the startup drive used by the operating system for temporary files. You can set the destination to be the Temporary Folder using the notation ".T"

On Macintosh the Temporary Folder is invisible, which can make it difficult to access files exported to this location. For this reason, it is a good idea to have your script check which platform the file is running on before setting the destination to the Temporary Folder.

Location of an open FileMaker database (Mac only)
You can specify the location of any currently open database. To do this, you must use the following notation:

".F:" and the filename

For example, ".F:PictureDB.fp7" will look for an open FileMaker database file named "PictureDB.fp7", and set the destination to its containing folder. To make the destination the same folder as the current file, use ".F:"&Get(FileName).

Network Note: When using the ".F" parameter for a file that is hosted over a network (by FileMaker Pro Server, for example), ExportFM will return a file not found error. But, if the file is simply being run across a network without being hosted by another computer (the file simply resides on another computer that is mounted on your Mac's desktop), then the pathway will be returned correctly.

User selected folder
You can let the user select the destination folder with the following parameter:

".U:" and the prompt statement

For example, ".U:Please select a destination folder" will bring up a dialog with the text "Please select a destination folder" that allows the user to select a folder.

Be aware that only a limited space is allowed for the text of your prompt statement.

On Macintosh --
A Select button appears in the location dialog.

On Windows --
ExportFM displays the Windows folder selection dialog. This dialog shows the hierarchical folder directory, but not applications or other files.

User selected file
This is similar selecting a folder, except that the user selects a specific file, and that file's name is then returned to your response field. You can do this with the following parameter:

".V:" and the prompt statement

For example, ".V:Please select a file" will bring up an open dialog with the text "Please select a file". The selected file's name will then be returned to the response field, and the destination folder will be updated to the enclosing folder.

Testing for Cancel:
If the user clicks the Cancel button in the select destination dialog, no destination will be set. Instead, ExportFM returns the response "$0:Cancel". To prevent errors occurring in your scripts, it is a good idea to always evaluate the response field immediately following a "user select" set destination step. You can use an If statement such as:

```
If [Table::Response Field = "$0:Cancel"]
```

or, more generically,

```
If [not IsEmpty(Table::Response Field)]
```

3. Relative Path

Once a destination folder is set, you can call Export-SetDestinationFolder again and specify a new destination relative to the current one. Note that in order to set the destination folder relative to the current destination folder, your parameter must begin with a directory separator (":" on Mac, "\" on Windows), otherwise ExportFM will assume you are resetting the full path.

Macintosh example--
If the destination is set to "Macintosh HD:FileMaker Docs:", you can next specify ":Backups" to enter deeper into enclosed folders.

Windows example--
If the destination is set to "C:\FileMaker Docs\", you can next specify "\Backups".

Going up one level
You can also go up one level relative to the current destination folder using a double colon "::" on Mac or a double period ".." on Windows.

Macintosh example 1--
If the destination is set to ".A", you can next specify "::" to go up one level from the FileMaker folder to, say, the Applications folder.

Macintosh example 2--
Or you could specify "::MS Word" to go up one level (again, to the Applications folder) and then select a different folder named "MS Word" as the destination folder.

Windows example--
If the destination is set to "C:\Program Files\FileMaker Docs\PictureDB", you could specify "..Extra Pictures" to go up one level (to FileMaker Docs) and then select a different folder named "Extra Pictures" as the destination folder.

Parameters:

FilePath – The location or file path where ExportFM will perform its next Export or NewFolder command.

Examples:

Set destination to be the desktop--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
```

Set destination to the folder containing the FileMaker application--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".A")]
```

Set destination to the same folder as the current database file (Mac only)--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".F:" &
    Get(FileName))]
```

Set destination to the same folder as an open database file named "SecondFile.FP7". (This file must be open but it doesn't need to be the file running the script)--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder";
    ".F:SecondFile.FP7")]
```

Let the user select a folder with the words "Store export file here"--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".U:Store export
    file here")]
If [Table::Response Field = "$0:Cancel"]
        Halt Script
End If
```

Let the user select a file with the words "Select a file", storing the file name in a field named 'Filename'--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".V:Select a
    file")]
If [Table::Response Field = "$0:Cancel"]
        Halt Script
Else
        Set Field ["Filename", Table::Response Field]
End If
```

Set destination to the folder "Saved Items" which is in the same folder as the FileMaker Pro application--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".A"]
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ":Saved
    Items")]
```

On a Mac, set destination to one folder UP from the FileMaker application's folder--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".A")]
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; "::")]
```

On a Windows machine, set destination to one folder UP from the Windows folder--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".W")]
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; "..")]
```

On a Mac, go up one folder from the current file's location, and create a new folder named "Website"--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".F:" & Get
    (FileName) & ".fp7")]
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; "::")]
Set Field[Table::Response Field; External("Export-NewFolder"; "Website")]
```

To check for an error after setting the destination folder using the full path method--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; "Macintosh
    HD:Desktop Folder:Exported Files:)]
If [Left(Table::Response Field), 1) = "$"]
    Show Message ["There is a problem with this location."]
    Halt Script
End If
```

Export-SetSourceFolder

External ("Export-SetSourceFolder", "FilePath")

Works just like Export-SetDestinationFolder. Sets the source folder for the Export-CreateShortcut, Export-MoveFile and Export-CopyFile commands.

This function works just like the Export-SetDestinationFolder command, using the same syntax and parameters. (See Export-SetDestinationFolder for details of how to set parameters.)

Export-SetSourceFolder is used in the following functions:

Export-CreateShortcut
Export-MoveFile
Export-CopyFile

For the above functions, Export-SetSourceFolder is used to specify the original or source location of the file to be moved or copied.

Export-SetDestinationFolder is used to specify where the file will be moved to or copied to.

Parameters:

FilePath – The location or file path where ExportFM will perform its next Export or NewFolder command.

Examples:

Set the source location to be the desktop--

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; ".D")]
```

Set source to the same folder as the current database file (Mac only)--

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; ".F:" &
    Get(FileName))]
```

Let the user select a folder with the words "Location of file to be copied"--

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; ".U:Location of file
    to be copied")]
If [Table::Response Field = "$0:Cancel"]
        Halt Script
End If
```

Let the user select a file with the words "File to be moved", storing the file name in a field named 'Filename'--

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; ".V:File to be
    moved")]
If [Table::Response Field = "$0:Cancel"]
        Halt Script
Else
        Set Field ["Filename", Table::Response Field]
End If
```

To check for an error after setting the source folder using the full path method--

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; "Macintosh
    HD:Desktop Folder:Exported Files:)]
If [Left(Table::Response Field), 1) = "$"]
        Beep
        Show Message ["There is a problem with this location."]
        Halt Script
End If
```

Export-GetDestinationFolder

External ("Export-GetDestinationFolder", "")

Lets you check the complete file path to ExportFM's current destination folder. This can be used for debugging or to store the file path in a field for future reference.

The Export-GetDestinationFolder function can be a valuable way to store the complete file path to a regularly referenced location, such as an export folder. It can also be useful to confirm the path which has been set by Export-SetDestinationFolder.

Caution: Problems can arise with a saved file path if, for example, one of the folders is moved or renamed. When possible, it is more reliable to save a path relative to a known folder, such as the desktop, FileMaker application, or the active database file.

Errors:

No errors are returned.

Parameters:

No parameter is used. Use empty quotes "".

Examples:

To simply check the filepath to the current destination folder--

```
Set Field[Table::Response Field; External("Export-GetDestinationFolder"; "")]
```

To let the user choose the destination folder and then store the file path in a field named "File Path"--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".U:Choose a
    destination folder")]
Set Field[Table::File Path; External("Export-GetDestinationFolder"; "")]
```

To calculate the folder name for the current destination folder and set it into a field named "Folder Name" (on a Mac)--

```
Set Field[Table::Response Field; External("Export-GetDestinationFolder"; "")]
Set Field[Table::Folder Name; Middle(Table::Response Field; Position(Table::Response
    Field; ":"; 1; PatternCount(Table::Response Field; ":") - 1) + 1;
    Length(Table::Response Field) - Position(Table::Response Field; ":"; 1;
    PatternCount(Table::Response Field; ":") - 1) - 1)]
```

To calculate the folder name for the current destination folder and set it into a field named "Folder Name" (on a Windows)--

```
Set Field[Table::Response Field; External("Export-GetDestinationFolder"; "")]
Set Field[Table::Folder Name; Middle(Table::Response Field; Position(Table::Response
    Field; "\"; 1; PatternCount(Table::Response Field; "\") - 1) + 1;
    Length(Table::Response Field) - Position(Table::Response Field; "\"; 1;
    PatternCount(Table::Response Field; "\") - 1) - 1)]
```

Export-GetSourceFolder

External ("Export-GetSourceFolder", "")

Identical to Export-GetDestinationFolder, but the file path returned is for ExportFM's current source folder, not the destination folder. This can be used for debugging or to store the file path in a field for future reference.

The Export-GetSourceFolder function can be useful to confirm the path which has been set by Export-SetSourceFolder.

Caution: Problems can arise with a saved file path if, for example, one of the folders is moved or renamed. When possible, it is more reliable to save a path relative to a known location, such as the desktop, FileMaker application, or the active database file.

Errors:

No errors are returned.

Parameters:

No parameter is used. Use empty quotes "".

Examples:

To simply check the filepath to the current source folder--

```
Set Field[Table::Response Field; External("Export-GetSourceFolder"; "")]
```

To let the user choose the source folder and then store the file path in a field named "File Path"--

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; ".U:Choose a source folder")]
Set Field[Table::File Path; External("Export-GetSourceFolder"; "")]
```

To calculate the folder name for the current source folder (by parsing the full path) and set it into a field named "Folder Name" (on a Mac)--

```
Set Field[Table::Response Field; External("Export-GetSourceFolder"; "")]
Set Field[Table::Folder Name; Middle(Table::Response Field; Position(Table::Response
    Field; ":"; 1; PatternCount(Table::Response Field; ":") - 1) + 1;
    Length(Table::Response Field) - Position(Table::Response Field; ":"; 1;
    PatternCount(Table::Response Field; ":") - 1) - 1)]
```

To calculate the folder name for the current source folder (by parsing the full path) and set it into a field named "Folder Name" (on Windows)--

```
Set Field[Table::Response Field; External("Export-GetSourceFolder"; "")]
Set Field[Table::Folder Name; Middle(Table::Response Field; Position(Table::Response
    Field; "\"; 1; PatternCount(Table::Response Field; "\") - 1) + 1;
    Length(Table::Response Field) - Position(Table::Response Field; "\"; 1;
    PatternCount(Table::Response Field; "\") - 1) - 1)]
```

Export-CopyFile

External ("Export-CopyFile", "CopyFile")

Copies the specified file from the source folder to the destination folder.

Export-CopyFile creates a copy of the specified file found in the source folder and places it in the destination folder. Unlike Export-MoveFile, Export-CopyFile leaves the original copy of the file in its source location.

As long as the file is located in the source folder, ExportFM will copy the file for you.

Errors:

If the file can't be found in the source folder, ExportFM will return an error.

An error is also returned if an identically named file already exists in the destination folder.

Parameter:

FileName – Any file with this filename will be copied from ExportFM's source folder to the destination folder.

Examples:

To copy a file named "CloneMe.tab" from a folder named "Stuff" to the desktop--

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; "C:\Stuff\")]
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field[Table::Response Field; External("Export-CopyFile"; "CloneMe.tab")]
```

(See details on the Export-SetDestinationFolder and Export-SetSourceFolder functions to understand how to set locations.)

To allow the user to select the file to be copied and set its new location--

```
Set Field[Table::Response Field; External("Export-SetSourceFolder"; ".V:Select the file to
    be copied")]
If [Table::Response Field = "$:Cancel"]
    Halt Script
Else
    Set Field [Table::Filename Field; Table::Response Field]
End If
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".U:Select the
    new location folder")]
If [Table::Response Field = "$:Cancel"]
    Halt Script
End If
Set Field[Table::Response Field; External("Export-CopyFile"; Table::Filename Field)]
If [not IsEmpty (Table::Response Field)]
    Beep
    Show Message ["An error occurred when attempting to copy the file."]
End If
```

Export-ListVolumes

External (“Export-ListVolumes”, “”)

Returns a list of all mounted volumes.

The Export-ListVolumes function returns a list of all mounted volumes -- local hard drives, remotely mounted hard drives, loaded removable drives, etc. -- separated by pipe characters "|".

On the Mac platform, the volume list returned uses the names given to each volume:

Jane's Mac|Jeff's Drive|Zip Backups

On Windows, the volume letters are returned:

a:\|c:\|d:\|e:\|f:\|g:\|m:\

Export-ExtractParameter

The Export-ExtractParameter command can be used to easily pick out a specific volume name from the volume list.

Parameters:

No parameter is used. Use empty quotes "".

Examples:

To get a list of all mounted volumes--

```
Set Field [Table::Response Field; External("Export-ListVolumes"; "")]
```

To use Export-ExtractParameter in order to get the number of mounted volumes--

```
Set Field [Table::Volume List; External("Export-ListVolumes"; "")]
Set Field [Table::Volume Total; External("Export-ExtractParameter"; "0|" & Table::
Volume List)]
```

To create a new record for each mounted volume, storing each volume name in a field called "Volume Name"--

```
Set Field [Table::Volume List; External("Export-ListVolumes"; "")]
Set Field [Table::Volume Total; External("Export-ExtractParameter"; "0|" & Table::
    Volume List)]
Set Field [Table::Global Counter; 1]
Loop
    New Record / Request
```

```
    Set Field [Table::Volume Name; External("Export-ExtractParameter";Table::Global
        Counter & "|" & Table::Volume List)]
    Exit Loop If [Table::Global Counter >= Table::Volume Total]
    Set Field [Table::Global Counter; Table::Global Counter + 1]
End Loop
```

Export-ListDestinationFolder

External ("Export-ListDestinationFolder", "ShowFolders?")

Returns a list of files in the destination folder. Gives you the ability to import all files or insert all pictures contained in a specific folder.

Export-ListDestinationFolder returns a list of files contained in the current destination folder, separated by pipe characters "|".

The file list returned will look like:

Graphic1.JPG|Graphic2.GIF|ƒGraphics Subfolder|ZZZ.txt

The file list follows the same order as they appear in the operating system's directory.

File Types

You can use the parameter to specify whether to only show standard files or to also list folders, aliases and hidden files.

To prevent confusion ExportFM adds a prefix to the file name that specifies the file type of each file:

"" (no prefix)	=	Standard File
ƒ	=	Folder
^	=	Alias
~	=	Hidden File
~*ƒ*	=	Hidden Folder

Export-ExtractParameter

The Export-ExtractParameter command can be used to easily pick out a specific file name from the file list. You can also loop through the file list to, for example, automatically insert all pictures contained in a folder.

Parameters:

ShowFolders? - Specifies whether to show only standard files or to also show folders, aliases and hidden files.

1 = Show all file types (folders, aliases and hidden files will have a file type prefix)

0 = Show only standard files

Examples:

To list all files on the desktop--

```
Set Field [Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field [Table::Response Field; External("Export-ListDestinationFolder"; "1")]
```

To list only the standard files on the desktop--

```
Set Field [Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field [Table::Response Field; External("Export-ListDestinationFolder"; "0")]
```

To list all files in a user specified folder--

```
Set Field [Table::Response Field; External("Export-SetDestinationFolder"; ".U:Select a
    folder")]
Set Field [Table::Response Field; External("Export-ListDestinationFolder"; "1")]
```

To use Export-ExtractParameter in order to get the number of files in the destination folder--

```
Set Field [Table::Response Field; External("Export-SetDestinationFolder"; ".U:Select a
    folder")]
Set Field [Table::File List; External("Export-ListDestinationFolder"; "1")]
Set Field [Table::File Total; External("Export-ExtractParameter"; "0|" & Table::File List)]
```

To create a new record for each standard file, storing each file's name in a field called "File Name"--

```
Set Field [Table::Response Field; External("Export-SetDestinationFolder"; ".U:Select a
    folder")]
Set Field [Table::File List; External("Export-ListDestinationFolder"; "0")]
Set Field [Table::File Total; External("Export-ExtractParameter"; "0|" & Table::File List)]
Set Field [Table::Global Counter; 1]
Loop
    New Record / Request
    Set Field [Table::File Name; External("Export-ExtractParameter"; Table::Global
        Counter & "|" & Table::File List)]
    Exit Loop If [Table::Global Counter >= Table::File Total]
    Set Field [Table::Global Counter; Table::Global Counter + 1]
End Loop
```

If you want to work with all file types but want to remove any file type prefixes from the file name, you can use the following calculation (in either a field calculation or a Set Field script step)--

```
Case(
Left(Table::File Name, 2) = "f~",
Right(Table::File Name, (Length(Table::File Name) - 2)),
Left(Table::File Name, 1) = "f" or Left(Table::File Name, 1) = "~" or Left(Table::File
    Name, 1) = "^",
Right(Table::File Name, (Length(Table::File Name) - 1)),
Table::File Name)
```

Export-NewFolder

External ("Export-NewFolder", "FolderName")

Lets you easily create a new folder inside ExportFM's current destination folder. This can be useful to create directories in which to export text files, image catalogs, even entire websites.

Note: The newly created folder becomes ExportFM's new destination folder.

When using the Export-NewFolder function, if a folder with the same name already exists in the destination folder, ExportFM changes its destination folder to be the specified folder without creating a new folder or replacing the existing folder.

Parameters:

FolderName - The name of the new folder to be created.

Example:

To create a destination folder "Exported Items" on the desktop--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field[Table::Response Field; External("Export-NewFolder"; "Exported Items")]
```

Export-DeleteFile

External ("Export-DeleteFile", "FileName")

Lets you delete any file that is located in ExportFM's destination folder.

Files that are locked or in use cannot be deleted. Folders cannot be deleted on the Windows platform.

Warning: Export-DeleteFile immediately and unrecoverably deletes the named file. It does not move it to the Trash, it does not check that it's the file you really meant (so be sure!)

Errors:

If the file does not exist, no error is returned.

If the file is locked or in use, the following error is returned:
"$-45:Could not delete file (bad spec, busy, protected)"

Parameter:

FileName – Any file with this specified filename will be permanently deleted from ExportFM's current destination folder.

Example:

To delete a file named "TestFile" located on the desktop--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field[Table::Response Field; External("Export-DeleteFile"; "TestFile")]
```

(See details on the Export-SetDestinationFolder function to understand how it is used to set the location to be the desktop.)

Export-Open

External ("Export-Open", "FileName")

Opens the specified file in the destination folder. This can be any type of file. ExportFM launches the necessary application – as determined by the file extension or, on Mac, the file type.

Export-Open will launch the document or application named in the parameter if it is found in the destination folder.

Note:

If you are opening a FileMaker Pro file, FileMaker will attempt to complete the current script before opening the new file. This may cause problems if there are several script steps remaining, or if the steps cause delays, such as with Show Message or Pause / Resume Script.

When opening a non-FileMaker file, the new application will become the active application window. If the script has remaining steps to perform, FileMaker will attempt to complete the script in the background.

For these reasons, the Export-Open step should normally be the last step performed by the script.

Errors:

If the file can't be found in the destination folder, ExportFM will return an error.

Parameter:

FileName – Any file with this filename in the destination folder will be launched.

Examples:

To open a FileMaker database with the file name entered into the Database Name field--

```
Set Field[Table::Response Field; External("Export-Open"; Table::Database Name)]
```

To open a file named Info.PDF located on the desktop--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field[Table::Response Field; External("Export-Open"; "Info.PDF")]
```

(See details on the Export-SetDestinationFolder function to understand how to set locations.)

To launch a referenced letter with the file name calculated in the Contact Letter Name field with the file path stored in a field named Letters Filepath--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; Table::Letters
    Filepath)]
Set Field[Table::Response Field; External("Export-Open"; Table::Contact Letter Name)]
```

To let a user select the file to be opened--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".V:Select the
    file to be opened")]
If [Table::Response Field = "$:Cancel"]
    Halt Script
Else
    Set Field [Table::Filename Field; Table::Response Field]
End If
Set Field[Table::Response Field; External("Export-Open"; Table::Filename Field);]
```

Information Functions

Export-GetTypes

External ("Export-GetTypes", "")

Returns the available picture types of the graphic image on the clipboard.

The Export-GetTypes function returns a list of the types of graphics in the clipboard. The types are separated by the pipe "|" character.

For example, when a GIF image is copied to the clipboard, Export-GetTypes returns "gif|pct".

Some types may be listed that ExportFM does not understand how to export.

Errors:

$xxxx:Could not access clipboard
This is usually because of insufficient memory to load clipboard file.

Parameters:

No parameter is used. Use empty quotes "".

Examples:

To copy a stored image to the clipboard and check which image types are available--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-GetTypes"; "")]
```

To see if a GIF is among the image types available on the clipboard--

```
Set Field [Table::Response Field; External("Export-GetTypes"; "")]
If [PatternCount (Table::Response Field; "GIF") > 0]
        Show Message ["There is a GIF on the clipboard."]
End If
```

To set the types into a dynamic pop-up menu--

```
Set Field [Table::Types Value List; External("Export-GetTypes"; "")]
Comment [SUBSTITUTE CARRIAGE RETURNS FOR PIPECHARACTERS "|"]
Set Field [Table::Types Value List; Substitute (Table::Types Value List; "|"; "¶")]
```

```
Comment [THE FIELD FORMAT FOR THE "SELECT TYPE" FIELD IS SET TO POP-UP MENU.]
Comment [THE VALUE LIST MUST BE DEFINED TO USE THE VALUES FOUND IN THE
    ABOVE "TYPES VALUE LIST" FIELD.]
Go to Field [Table::Select Type]
```

Export-GetFileInfo

External("Export-GetFileInfo","FileName")

Returns the 3 or 4 character file type of the specified file, if it is located in the destination folder.

The Export-GetFileInfo function returns the 3 or 4 character file type of a saved file.

Use Export-SetDestinationFolder to specify the location of the file to be used. Set the file name as the Export-GetFileInfo parameter.

When checking the file type of a shortcut or alias file, ExportFM returns the file type of the original file the shortcut refers to.

Additional Dividers

Export-GetFileInfo returns a value such as

```
gif||
```

The first segment of the response is the file type. The two pipe characters are placeholders that are currently unused. A future version of ExportFM will return additional information, using the pipe characters as separators.

Export-ExtractParameter

Use the Export-ExtractParameter function to easily grab the file type information returned by Export-GetFileInfo.

```
Set Field[Table::Response Field; External("Export-ExtractParameter"; "1|" &
    External("Export-GetFileInfo"; "Test File"))]
```

(For more information on how to use this function, see the documentation on Export-ExtractParameter.)

Using with Insert Picture and Insert Movie

Used in conjunction with Export-ListDestinationFolder, Export-GetFileInfo gives you the ability to create a script that recognizes all of the image files within a folder (or even images of a certain type, such as JPEGs), and then insert only those images into container fields, using Export-CreateShortcut.

Or, suppose you have a folder containing mixed media with image files and QuickTime movies, and you want to store them all in FileMaker container fields. Images require the Insert Picture script step, but movies require the Insert Movie script step. Export-GetFileInfo allows your script to distinguish between the two file types and branch appropriately to call the correct script step.

Parameter:

FileName - If a file with this filename exists in ExportFM's current destination folder, its 3 or 4 character file type code will be returned.

Examples:

To get the file info of a file named "Test File" on the desktop--

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".D")]
Set Field[Table::Response Field; External("Export-GetFileInfo"; "Test File")]
```

To get the file info of a user-specified file--

```
Set Field[Table::File Name; External("Export-SetDestinationFolder"; ".V:Select a file")]
Set Field[Table::Response Field; External("Export-GetFileInfo"; File Name)]
```

To extract the file type of a user-specified file--

```
Set Field[Table::File Name; External("Export-SetDestinationFolder"; ".V:Select a file")]
Set Field[Table::Response Field; External("Export-ExtractParameter"; "1|" &
    External("Export-GetFileInfo"; Table::File Name))]
```

Export-GetImageInfo

External("Export-GetImageInfo", "ImageType")

Returns the characteristics -- height, width, bit depth, compression quality, resolution, file size and image type -- of an image on the clipboard.

Response

GetImageInfo returns a text string containing eight items separated by pipe characters "|":

"Vert|Horiz|BitDepth|Quality|V-Resolution|H-Resolution|FileSize|ImageType"

Vert - The vertical dimension (in pixels) of the image.

Horiz - The horizontal dimension (in pixels) of the image.

BitDepth - The color or gray-scale bit depth for the resulting image. Smaller bit depths give more compact images that load and display faster and require less storage space, but they have fewer colors or gray-scale range.

1 = Color - Bit Depth 1
2 = Color - Bit Depth 2
4 = Color - Bit Depth 4
8 = Color - Bit Depth 8
16 = Color - Bit Depth 16
24 = Color - Bit Depth 24
32 = Color - Bit Depth 32
33 = Gray-Scale - Bit Depth 1
34 = Gray-Scale - Bit Depth 2
36 = Gray-Scale - Bit Depth 4
40 = Gray-Scale - Bit Depth 8

Quality - The quality of the image compression. Higher compression levels yield smaller image files which load and display faster, but have lower image quality.

0 = Minimum Quality, Maximum Compression
256 = Low Quality, High Compression
512 = Normal Quality, Medium Compression
768 = High Quality, Low Compression
1023 = Maximum Quality, Minimum Compression
1024 = Lossless Quality

V-Resolution - The vertical resolution of the image in dots per inch. The standard for screen display is 72. Lower resolution yields a smaller image file that is coarser visually.

H-Resolution - The horizontal resolution of the image in dots per inch. The standard for screen display is 72. Lower resolution yields a smaller image file that is coarser visually.

FileSize - The image's file size in bytes.

ImageType - The format of the image.

These eight pieces of information roughly correspond to the parameters used by ExportFM's ConvertImage command.

QuickTime

Export-GetImageInfo requires QuickTime 4.0 or greater. If you do not already have this installed on your computer, it can be downloaded for free at <http://www.apple.com/quicktime/download/>.

If QuickTime is not installed, ExportFM will return an error. You can use ExportFM's CheckQT command to verify that QuickTime 4 or greater is installed.

Caution: QuickTime occasionally reports incorrect values. For example, if you create and save a low quality JPEG, QuickTime reports it as high quality, even though it has

clearly been converted to a high compression level with loss of quality. QuickTime also occasionally misreports the pixel depth.

Parameters:

ImageType – the three character type code of the image on the clipboard.

Examples:

To get image information for a GIF already in the clipboard--

```
Set Field [Table::Response Field; External("Export-GetImageInfo";"gif")]
```

To copy a PICT image to the clipboard and check the image information--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-GetImageInfo";"pct")]
```

Since the GetImageInfo command requires a recent version of QuickTime to be installed and active, it is a good idea to first check to make sure QuickTime is available. (See Export-CheckQT for more details.)--

```
If [External("Export-CheckQT";"") <>1]
        Show Message ["You must have QuickTime 4.0 or greater installed to run this
            script."]
        Halt Script
End If
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-GetImageInfo";"jpg")]
```

To parse the GetImageInfo data into separate fields--

```
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-GetImageInfo";"gif")]
Set Field [Table::Vert; MiddleWords(Table::Response Field; 1; 1)]
Set Field [Table::Horiz; MiddleWords(Table::Response Field; 2; 1)]
Set Field [Table::Bit Depth; MiddleWords(Table::Response Field; 3; 1)]
Set Field [Table::Quality; MiddleWords(Table::Response Field; 4; 1)]
Set Field [Table::V Res; MiddleWords(Table::Response Field; 5; 1)]
Set Field [Table::H Res; MiddleWords(Table::Response Field; 6; 1)]
Set Field [Table::File Size; MiddleWords(Table::Response Field; 7; 1)]
Set Field [Table::Image Type; MiddleWords(Table::Response Field; 8; 1)]
```

Export-GetRefName

External("Export-GetRefName","")

Returns the file name of the referenced image or other object copied from a container field to the clipboard.

Export-GetRefName lets you check the name of the original image, movie, or other object file that is stored by reference in a container field.

Copy Script Step

The contents of the container field must first be copied to the clipboard before the Export-GetRefName can be used. Thus, the essential two script steps are:

Copy [Select; Table::Container Field]
SetField [Table::Response Field; External("Export-GetRefName";"")]

Broken File References

If the referenced file cannot be found in its expected location, ExportFM will still attempt to return the originally referenced file name.

The FileMaker Pro application will return an error, however, if your script attempts to copy the contents of a container with a broken file reference. If you want to check the expected filepath when the image is no longer found, you must use Set Error Capture [On] to suppress the FMP error.

Export-GetRefName does not resolve broken references, check that the path is still valid, nor verify that the source file still exists.

Errors:

If you attempt to use this function when copying an image to the clipboard that is not stored by reference, ExportFM returns an error, such as:

$-102:The requested type is not in the clipboard
$0:The requested type is not in the clipboard

Parameter:

No parameter is used. Use empty quotes "".

Example:

To get the file name of a file stored by reference in a field named Image Container--

Copy [Select; Table::Image Container]
SetField [Table::Response Field; External("Export-GetRefName";"")]

Export-GetRefPath

External("Export-GetRefPath", "")

Returns the file path of the referenced image or other object copied from a container field to the clipboard.

Export-GetRefPath gives you the ability to get the file path to the original image, movie, or other object file that is stored by reference in a container field.

Copy Script Step

The contents of the container field must first be copied to the clipboard before the Export-GetRefPath can be used. Thus, the essential two script steps are:

Copy [Select; Table::Container Field]
SetField [Table::Response Field; External("Export-GetRefPath"; "")]

Filepath Response Format

If the image or object copied to the clipboard is stored in its container field by reference, ExportFM will return the filepath in one of two formats, depending on which platform is being used:

Windows: C:\WINDOWS\Desktop\
Mac OS X: Macintosh HD:Users:userfile:Desktop:

Broken File References

If the referenced file cannot be found in its expected location, ExportFM will still attempt to return the originally referenced file path.

The FileMaker Pro application will return an error, however, if your script attempts to copy the contents of a container with a broken file reference. If you want to check the expected filepath when the image is no longer found, you must use Set Error Capture [On] to suppress the FMP error.

Export-GetRefPath does not resolve broken references, check that the path is still valid, nor verify that the source file still exists.

Errors:

If you attempt to use this function when copying an image to the clipboard that is not stored by reference, ExportFM returns an error, such as:

$-102:The requested type is not in the clipboard
$0:The requested type is not in the clipboard

Parameter:

No parameter is used. Use empty quotes "".

Example:

To get the file path of a file stored by reference in a field named Image Container--

```
Copy [Select; Table::Image Container]
SetField [Table::Response Field; External("Export-GetRefPath";"")]
```

Export-GetMouseUp

External("Export-GetMouseUp","")

Returns the pointer coordinates when the mouse button was last clicked. Set an entire layout or container field as a single button and know precisely where the user has clicked.

Export-GetMouseUp returns the mouse pointer's horizontal and vertical coordinates from the last 'mouse up' -- that is, the last time the mouse button was clicked.

The response of Export-GetMouseUp contains four values separated by pipe characters "|". The response might look like "20|45|221|300". The first two values are the horizontal and vertical coordinates relative to the front window's upper left corner. The 3rd and 4th values are relative to the bottom right corner.

So, if the response is:

20|45|221|300

20 = Horizontal, from upper left corner
45 = Vertical, from upper left corner
221 = Horizontal, from bottom right corner
300 = Vertical, from bottom right corner

If the mouse has not yet been clicked, Export-GetMouseUp returns "-1|-1|-1|-1".

OS X Note:
On Mac OS X, the coordinates returned by Export-GetMouseUp are the current mouse pointer's position, not the position when the mouse button was last clicked. In normal usage (when the mouse click triggers the script that contains the Export-GetMouseUp function), the current mouse position is the same or nearly the same as the last clicked position. If you have several script steps before Export-GetMouseUp, however, and if the user is still moving the mouse, then a discrepancy can occur. For this reason, Export-GetMouseUp should be in the first script step.

Parameters:

No parameter is used. Use empty quotes "".

Example:

To get the pointer coordinates the last time the mouse was clicked--

Set Field[Table::Response Field;External("Export-GetMouseUp";"")]

Export-ExtractParameter

External ("Export-ExtractParameter", "Parameter#|TestString")

Quickly calculates the nth parameter of a multi-parameter test string.

Export-ExtractParameter views the first parameter ("Parameter#") on its own; all subsequent parameters divided by pipe characters "|" are seen as a separate Test String.

The first parameter tells ExportFM which of the subsequent parameters to extract.

For example, if the Export-ExtractParameter string is "3|Joe|Fred|Nancy|Susan", ExportFM will return "Nancy", the third parameter.

If the Parameter # specifies a number higher than the number of parameters in the Test String, Export-ExtractParameter returns the last one. If the Parameter Number is zero "0" or a negative number, the number of parameters is returned instead.

Errors:

Export-ExtractParameter never returns an error.

Parameters:

Parameter# - The parameter number to be extracted from the subsequent TestString.

TestString - The multi-parameter string from which to extract the specified parameter number. The parameters contained in the TestString must be divided by pipe characters "|".

The parameters must be separated by a pipe character "|".

Examples:

To get the second parameter in the string "Left|Right|Up|Down"--

```
Set Field[Table::Response Field; External("Export-ExtractParameter";
    "2|Left|Right|Up|Down")]
```

To get the third parameter in the string contained in a field named Response String--

```
Set Field[Table::Response Field; External("Export-ExtractParameter"; "3|" & Response
    String)]
```

To get the total number of parameters separated by pipe characters "|" contained in a field named Response String--

```
Set Field[Table::Response Field; External("Export-ExtractParameter"; "0|" & Response
    String)]
```

Export-CheckQT

External ("Export-CheckQT", "")

Tests whether QuickTime 4.0 or greater is installed. ExportFM's ConvertImage and GetImageInfo commands require a recent version QuickTime to work properly.

QuickTime 4.0 or greater must be installed to use ExportFM's ConvertImage and GetImageInfo commands. If you try to use either of those commands QuickTime, ExportFM will return an error.

The CheckQT command allows you to make sure an appropriate version of QuickTime is installed before calling ConvertImage or GetImageInfo. If QuickTime is not installed, you can alert the user.

Response

CheckQT returns either a "1" or "0".

1 = QuickTime 4.0 or greater is installed
0 = Either an earlier version of QuickTime is installed, or QuickTime is not installed at all. In that case, ConvertImage and GetImageInfo will not work.

Parameters:

No parameter is used. Use empty quotes "".

Examples

To check if QuickTime 4.0 or greater is active--

```
If [External("Export-CheckQT"; "") <>1]
    Show Message ["You must have QuickTime 4.0 or greater installed to run this
        script."]
```

```
        Halt Script
End If
```

To check if QuickTime is available before using the ConvertImage function. (See Export-ConvertImage for more details.)--

```
If [External("Export-CheckQT";"") <>1]
        Show Message  ["You must have QuickTime 4.0 or greater installed to run this
            script."]
        Halt Script
End If
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-ConvertImage";"pct|80|80||256|")]
```

To check if QuickTime is available before using the GetImageInfo function. (See Export-GetImageInfo for more details.)--

```
If [External("Export-CheckQT";"") <>1]
        Show Message  ["You must have QuickTime 4.0 or greater installed to run this
            script."]
        Halt Script
End If
Copy [Select; Table::Graphic Container Field]
Set Field [Table::Response Field; External("Export-GetImageInfo";"jpg")]
```

Register Functions

Export-Register

External("Export-Register", "LicenseeName|MacRegistration|WinRegistration")

Registers the ExportFM plug-in. This function can also be used to verify that the plug-in is active and which version is being used.

Important: If unregistered, ExportFM will run in a limited Demo mode and cease to operate after 30 minutes.

Storage of Registration Codes:

In our demo files, we have you enter your registration codes into fields where they are stored. This is the most convenient for you, the user. However, from a security point of view, it is safer to enter the codes directly into the Export-Register script step. You can, for example, have a single script step that doesn't call on data stored in fields:

```
Set Field [Table::Registration Response; External("Export-Register";
    "LICENSEENAME|MACREGISTRATIONCODE|WINREGISTRATIONCODE")]
```

Important: You must register ExportFM EVERY TIME you launch FileMaker Pro. A simple way to do this is to include a registration script step in your solution's On Open script. If you have multiple files in your database, you should either:

1) Include the registration step in the On Open script for each database, or
2) Require that the primary file be launched to ensure that its On Open script will register ExportFM.

An alternate approach is to simply re-register ExportFM at the beginning of any script that uses an ExportFM function.

Make sure your registration response field is a text or global text field

If you make the registration response field a number or global number field, and then register ExportFM, the correct registration text will be sent to the number field ("Registered ExportFM..."). However, if you then check the number field for the registration text, FileMaker will not see the text -- only the numerals of the version number in the text. To prevent this from happening, check your field definitions and make sure that the field you are using to store ExportFM's registration response is a text or global text field.

Responses:

A successful registration returns--
Registered ExportFM (version #)

If no match between the registration code(s), the licensee name and the platform --
Invalid registration ExportFM (version #)

Parameters:

LicenseeName – The user's name. This name must appear exactly as it does in the registration confirmation you received when you purchased the plug-in.

MacRegistration – The registration code used when running on the Macintosh platform.

WinRegistration – The registration code used when running on the Windows platform.

The parameters must be separated by a pipe character "|".

Examples:

To simply register ExportFM--

```
Set Field [Table::Registration Response Field; External ("Export-
    Register",;"LICENSEENAME|MACREGISTRATIONCODE|WINREGISTRATIONCODE")]
```

To check if ExportFM is installed and registered and, if not, quit FileMaker--

```
Set Field [Table::Registration Response Field; External ("Export-Register";
    "LICENSEENAME|MACREGISTRATIONCODE|WINREGISTRATIONCODE")]
If [not PatternCount (Table::Register Response Field, "Registered")]
        Show Message ["ExportFM is not active and is required to run this file."]
        Quit Application
End If
```

Export-Version

External ("Export-Version", "")

The Export-Version function returns just the version number of ExportFM without any other text. This makes it easy to determine the version of ExportFM currently in use.

```
External ("Export-Version", "")
```

No parameter is used.

The response returns only ExportFM version number, such as:

7.0.2

Parameters:

No parameter is used. Use empty quotes "".

Examples:

To check the version number of ExportFM--

```
Set Field [Table::Response Field; External("Export-Version"; "")]
```

To check if ExportFM is installed and registered and, if not, quit FileMaker--

```
Set Field [Table::Response Field; External("Export-Version"; "")]
If [Table::Response Field = "?" or IsEmpty(Table::Response Field)]
        Show Message ["ExportFM is not active and is required to run this file."]
        Quit Application
End If
```

Troubleshooting Guidelines

If you are encountering unexpected results when using ExportFM with your FileMaker solution, try the following troubleshooting steps to help pinpoint the source of the problem.

– Make sure there is only one version of ExportFM installed

Delete any earlier versions of ExportFM. Simply dragging the new plug-in into the correct folder will not assure that the old version will be replaced. Their names may be different.

– Make sure ExportFM is installed correctly

If ExportFM appears to not be doing anything, make sure the plug-in is installed correctly.

Place the ExportFM plug-in into FileMaker 7's " Extensions" folder.

Restart FileMaker after installing the plug-in.

Make sure ExportFM is selected in the Plug-Ins section of the Application Preferences (found in the Edit Menu).

– Restart FileMaker

If you are using an unregistered version of ExportFM, it ceases to function 30 minutes after FileMaker Pro is launched OR after 20 exports have been performed. To restore ExportFM functions before you are ready to purchase a user license, you must restart FileMaker Pro.

– Check the Registration Status

Make sure that ExportFM is actually installed and registered. You can do this by attempting to re-register ExportFM:

```
Set Field [Table::Registration Response Field; External ("Export-Register";
    "LICENSEENAME|MACREGISTRATIONCODE|WINREGISTRATIONCODE")]
```

If the response in the Registration Response Field is blank, ExportFM is not installed; in which case, none of the ExportFM dialog automation functions will work properly. You should first make sure that the ExportFM plug-in is placed into the System folder found within the FileMaker folder (on Windows), or in the FileMaker Extensions folder (on Mac).

After you have done that, go to the Application Preferences found in the Edit menu. Select the Plug-Ins tab and make sure that ExportFM is in the list and checked.

If the Registration Response Field is showing an "Invalid Registration", ExportFM is running in Demo mode. In Demo mode, file names may have the word "DEMO" added to them. Double check your registration codes if you have already purchased a user license.

Important: Register every time FileMaker is launched

When incorporating ExportFM into your own FileMaker solutions, you must register ExportFM EVERY TIME you launch FileMaker Pro. A simple way to do this is to include a registration script step in your solution's On Open script. If you have multiple files in your database, you should either:

1) Include the registration step in the On Open script for each database, or
2) Require that the primary file be launched to ensure that its On Open script will register ExportFM.

An alternate approach is to simply re-register ExportFM at the beginning of any script that uses an ExportFM function.

Make sure your registration response field is a text or global text field

If you make the registration response field a number or global number field, and then register ExportFM, the correct registration text will be sent to the number field ("Registered ExportFM..."). However, if you then check the number field for the registration text, FileMaker will not see the text -- only the numerals of the version number in the text. To prevent this from happening, check your field definitions and make sure that the field you are using to store ExportFM's registration response is a text or global text field.

For more information on registering ExportFM in your FileMaker solutions, please see the details on the Export-Register function.

– Check the ExportFM response.

In most cases, when an error occurs, ExportFM will send a response back during your Set Field step. You can pause your script and visually check ExportFM's response before proceeding.

For example:

```
Set Field [Table::Response Field; External("Export-Export"; "TestFile|txt|TV0D")]
Pause/Resume Script
```

Or, build an error check directly into your script:

```
Set Field[Table::Response Field; External("Export-Export", "TestFile|txt|TV0D")]
If [not IsEmpty (Table::Response Field)]
```

```
    Beep
    Show Message ["An error has occurred."]
    Halt Script
End If
```

– Check ExportFM's Destination

If ExportFM is having trouble locating files or seems to be saving or exporting them to incorrect locations, double check ExportFM's Destination Folder. After setting the destination, use DM-GetDestinationFolder to see the file path DialogMagic is using:

```
Set Field[Table::Response Field; External("Export-SetDestinationFolder"; ".A")]
Set Field[Table::File Path; External("Export-GetDestinationFolder"; "")]
```

Or, use Export-GetDestinationFolder just prior to performing its next task in order to see where the action will be performed:

```
Set Field[Table::File Path; External("Export-GetDestinationFolder"; "")]
Pause/Resume Script
Set Field[Table::Response Field; External("Export-Export"; "TestFile|txt|TV0D")]
```

In both of the above examples, the file path for ExportFM's destination folder will be set into a field named "File Path". Make sure that this file path is the path you expected. If not, you may need to double check your Export-SetDestinationFolder step.

– Make Sure QuickTime 4.0 or Greater is Installed

Errors will occur with the ConvertImage and GetImageInfo commands if QuickTime 4.0 or greater is not installed. If you do not already have this installed on your computer, it can be downloaded for free at <http://www.apple.com/quicktime/download/>.

Older versions of QuickTime allowed for a "minimum installation" without components necessary for some of ExportFM's functions. If you have an earlier version of QuickTime installed but you only have the minimum installation, the Export-CheckQT command may still return a 1 (indicating that QuickTime is installed), but errors may still result. If this is the case, you should uninstall QuickTime, and then reinstall the FULL set of QuickTime files. This is not a problem with the most recent versions of QuickTime.

– If ConvertImage does not work on Windows 2000 or XP

The ConvertImage command can encounter problems with certain BMP image files on Windows 2000 or XP. (This problem may, for example, arise during the Report Example in the ExportFM demo files.) If you are having problems with the ConvertImage command on Windows 2000 or XP, you may need to adjust the display settings for your monitor. In the Display control panel, go to the Settings tab and adjust the color resolution downward to 256 colors. Then try the ConvertImage command again to see if the problem is solved.

New Millennium
COMMUNICATIONS

New MillenniumCommunications
1332 PearlStreet
Boulder,CO 80302 USA
303-444-1476

plug-ins@nmci.com
www.newmillennium.com