launch.it 2

Tutorial

Updated: April 8th 2003

 www.dacons.net info@dacons.net

Table of Content

What is launch.it?

Welcome to launch.it 2

launch.it 2 enables you to store any file name and path in a FileMaker Pro field using a simple File Open dialog or automated import routines. The file and its parent application can be launched from within FileMaker.

Now you can manage any files with your FileMaker Pro solution. You can link certain files on your local computer or your intranet to records of your database.

For example: Store references to all corresponding files for one customer (i.e.: Word Documents) in your customer database. Easily find, open and organize documents related to that one customer.

In addition you can set filters to let users only select certain files when importing them to FileMaker.

launch.it 2 main features overview

1)
Select a file directly from FileMaker using a simple File Open dialog. Retrieve the file name as well as the complete file path.

2)
Retrieve a list of all files from a set folder, including subfolders and store them in FileMaker Pro fields. Link the content of project folders to FileMaker records.

3)
Open files and their parent applications from within FileMaker.

4)
Set filters for certain file types when importing file names and paths.

5)
Retrieve file icons for user-friendly presentation in FileMaker.

1. Installing launch.it 2

Decompress the package which you download from Dacons' web site and locate the FileMaker plug-in file (you will find it in the *Plug-In* folder). Please ensure that the FileMaker application is closed.

Mac OS users:
Place the plug-in *launch.it* in your *FileMaker Extensions* folder.

Windows users:
Place the plug-in *launch.it* in your FileMaker *System* folder.

Make sure you install the proper version for your operating system and ensure that you delete ALL previous versions of launch.it that are in your *System* folder (Windows) or in your *FileMaker Extensions* folder (Mac OS).

launch.it comes in two versions:

- Windows (Windows 98 and up including Windows XP)
- Mac (FAT binary for Mac OS Classic 8.6 and higher and Mac OS X)

The plug-in is compatible with FileMaker 4 through 6 (Pro, Developer and Runtime editions).

2. Step-by-step tutorial

This chapter gives easy to follow instructions on how to implement the launch.it functions into your new or existing FileMaker solution.

We suggest you open the FileMaker *Example File*, which is part of the package you downloaded. It will make it easy for you to follow the instructions.

2.1 Processing a single file

This section will explain how to:
- Set a file type filter for a File Open dialog
- Show a File Open dialog and import the selected file paths into a FileMaker field
- Get the file name according to the imported file paths
- Open the file and its parent application from within FileMaker Pro

Fields

The *Example File* mentioned above contains the following fields to achieve these steps:

Result	*Global text field* that contains the result returned by the plug-in's functions
File Filter	*Global text field* that contains the file type filter for the Open File dialog to be shown
File Path	*Text field* which will contain the path of the selected file
File Name	*Text field* which will contain the file name of the selected file

Scripts

The following paragraphs describe the main steps of the scripts ***Import Single File Path*** and ***Open File***. For an in-depth description please also refer to the chapter External Functions in this tutorial.
launch.it plug-in functions are highlighted in blue, names of the fields mentioned above will be printed red.

Import Single File Path script

```
Set Field ["Result", "External("Laun-SetDialogFilter", File Filter)"]
```

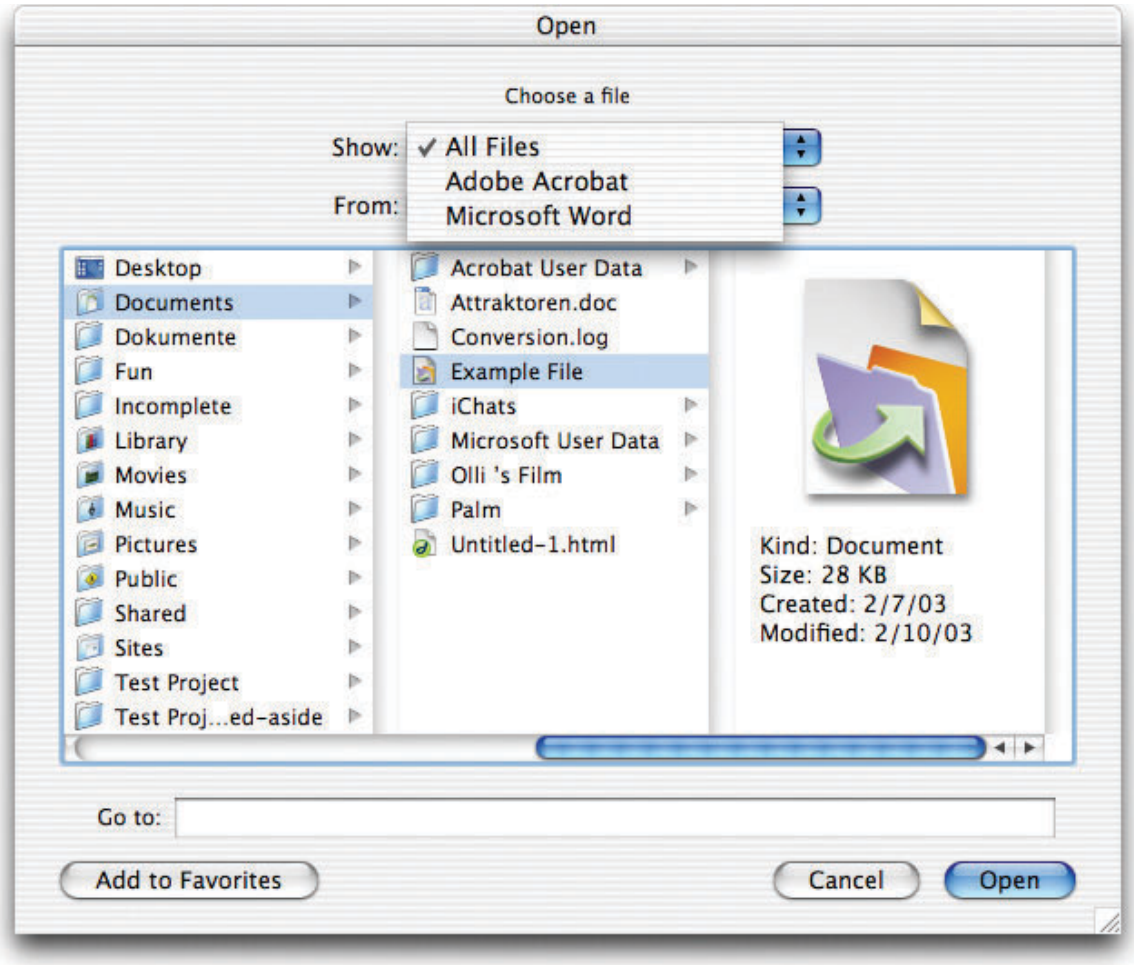
This script step sets the file type filter for the Open File dialog which is to be shown. The filter should be formatted like this:

Application;.extension*

For example: If the File Open dialog should only show files with the extension .pdf, the following would be set in the File Filter field: *Adobe Acrobat;*.pdf*

Multiple filters for the same File Open dialog can be set using a ';' between them. For Example:
Adobe Acrobat;.pdf; Microsoft Word;*.doc*
will show Acrobat or Microsoft Word files. The picture of the result on the next page.

To disable any filtering use: *.* In this case all files will be displayed in the File Open dialog.



The next script step shows the File Open dialog, which the file type filter has been set for in the previous step:

```
Set Field ["File Path", "External("Laun-GetFilePath", "")"]
```

The path of the selected file will be returned to the field File Path. You can also use a global text field instead to check if the user selects a file or hits the Cancel button in the File Open dialog which returns an empty result. This is demonstrated in the Example File.

In this case an empty parameter ("") is passed to the function Laun-GetFilePath. You can use the parameter for this function to set a custom title for the File Open dialog.

```
Set Field ["File Name", "External("Laun-GetFileName", File Path)"]
```

The next step returns the file name of the path which has been retrieved before. This file name is stored in the field File Name.

Open File script

To open the file that you selected above the Laun-OpenFile function should be called:

```
Set Field ["Result", "External("Laun-Open", File Path)"]
```

2.2 Processing multiple files in folders

This section will explain how to:
- Set a filter to import certain files with certain extension(s) from a folder
- Import multiple file paths from a folder, including subfolders

Fields

The launch.it Example File contains the following fields with regard to this task:

Result	*Global text field* that contains the result returned by the plug-in functions
Selected Folder	*Global text field* that contains the path of the selected folder which is to be processed
Folder Filter	*Global text field* which contains the filter for the file types to be imported from a certain folder
Imported Path	*Global text field* which contains file paths returned by the plug-in
File Path	*Text field* that contains the file paths of a file after it has been checked and copied to the database (as mentioned in the previous chapter)
File Name	*Text field* which contains the file (as mentioned in the previous chapter)

Scripts

In order to make reading easier launch.it plug-in functions are again highlighted in blue, names of the fields mentioned above will be printed red.

The *Example File* contains the script ***Import File Paths From Folder*** to demonstrate the import of multiple file paths from a folder. This chapter will describe the main launch.it functions used in this script.

Choose a folder for file paths import

```
Set Field ["Selected Folder", "External("Laun-ChooseFolder", "")"]
```

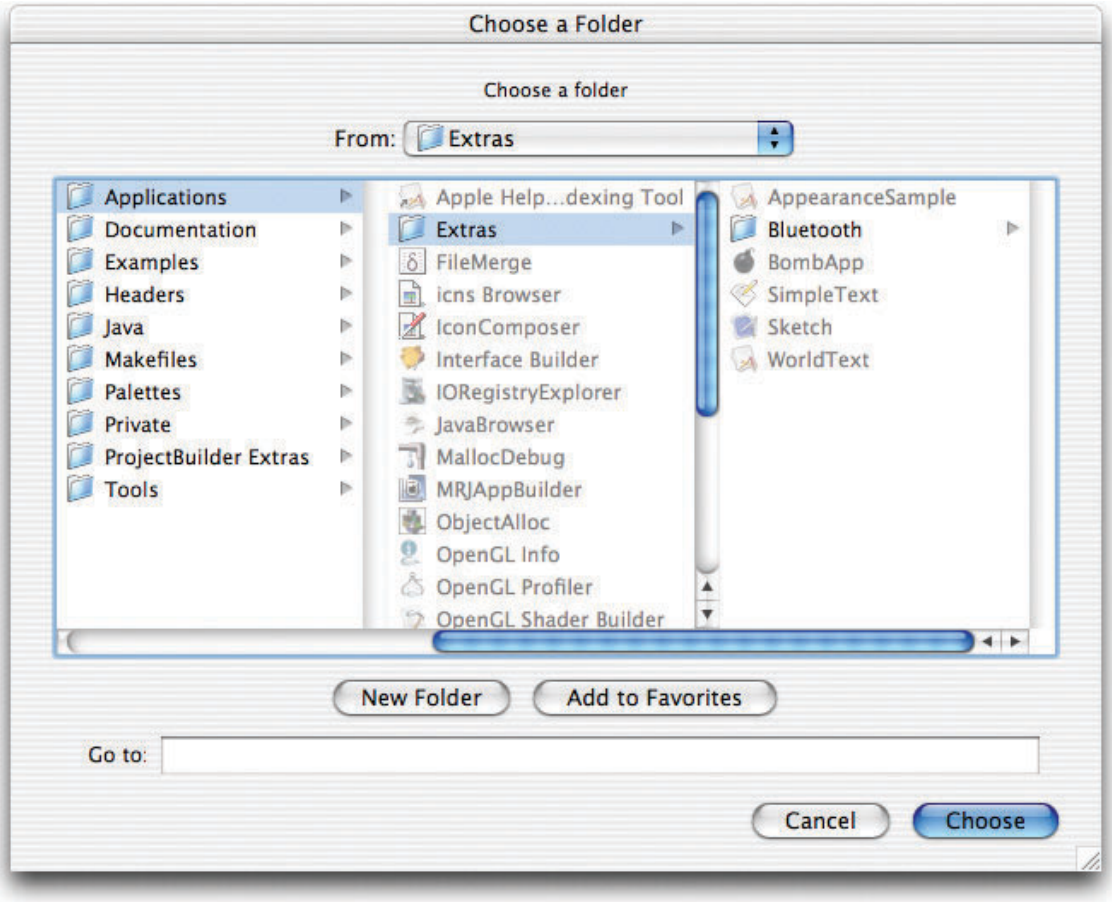
This script lets the user select a folder that should be processed. This can be any local or network folder which the user has read access to. (See screenshot next page)

Set file type filter for import

```
Set Field ["Result", "External("Laun-SetFileFilter", Folder Filter)"]
```

This script sets the filters for the file types to be imported from the selected folder.

For example: If you only would like to import file paths from a certain folder with the extension .pdf, the following should be set in the Folder Filter field: **.pdf*
If you would like to import files paths of several file types they have to be separated by ';'.
For example: **.pdf;*.xls*



Subfolder handling

```
Set Field ["Result", "External("Laun-UseSubfolders", "ON")"]
```

This step makes launch.it import file paths from subfolders. Use parameter “OFF” if you would NOT like to import subfolders. In the *Example File* the “ON” or “OFF” setting is stored in a global text field called Import Subfolders.

Pass selected folder to the plug-in

```
Set Field ["Result", "External("Laun-SetFolder", Selected Folder)"]
```

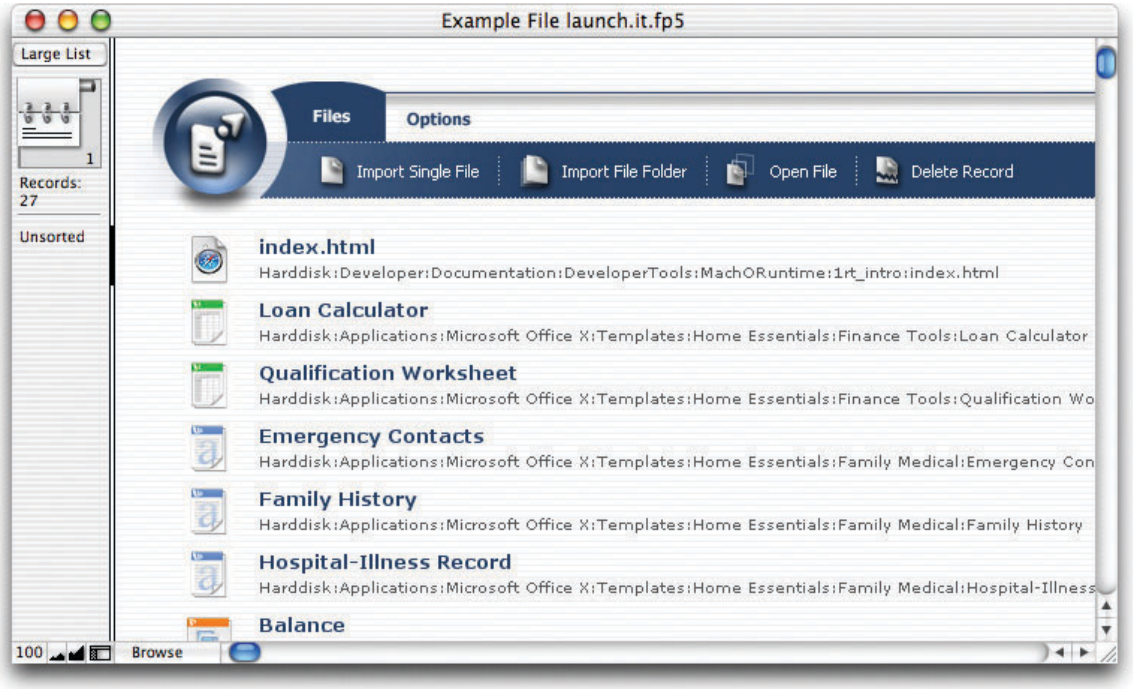
Use this script step to pass the path of the folder which has been selected using a Choose Folder dialog or any other folder path to the plug-in.

Import file paths from the specified folder

```
Enter Browse Mode []
Loop
   Set Field ["Imported Path", "External("Laun-GetNextFilePath","")"]
   Exit Loop If ["IsEmpty(Imported Path)"]
   New Record/Request
   Set Field ["File Path", "Imported Path"]
End Loop
```

This script loop is used to import all paths from the specified folder. launch.it will only return paths of file types according to the import filter which has been set before. If there is no further path to be imported the function Laun-GetNextFilePath returns an empty result. This is used for the *Exit Loop* condition.

As demonstrated in the *Example File* you can also extend the loop by importing the file name after retrieving the file path from the plug-in. Therefore the function Laun-GetFileName is used. Please review the previous chapter for details.



3. Importing Icons

The launch.it *Example File* shows how to import file icons. This is helpful if you would like to design a user friendly interface for your FileMaker solution.

launch.it provides two functions which enable you to copy a small and a large version of a file icon to a FileMaker *Container* field. Please note that these icons have to be inserted into the field using a *Paste* script step command. Therefore the target field of the Paste command has to be present in the layout which is currently shown.

The ***Import File Paths From Folder*** script contains the following script steps to import file icons:

```
Set Field [“Result”, “External("Laun-CopyLargeFileIcon", File Path)”]
Paste [“Large Icon”]
```

The first script step copies the file icon (32x32 pixels) of the file specified by the parameter File Path to the clipboard. The second function pasts the clipboard content which is the file icon into the container field Large Icon. Use the function Laun-CopySmallFileIcon instead to copy the small version of a file icon (16x16 pixels). The complete file path is needed as function parameter in any case.

Since the clipboard is the only way to import media content into FileMaker it makes sense to restore the user clipboard after using it to import file icons. This is done by the following script step:

```
Set Field [“Result”, “External("Laun-RestoreClipboard","")”]
```

Hint:
On Windows only standard clipboard content such as plain text or bitmaps can be restored using this function. Please note that the icons are stored as bitmaps in FileMaker container fields. This can increase the size of your database file significantly when importing many file icons.

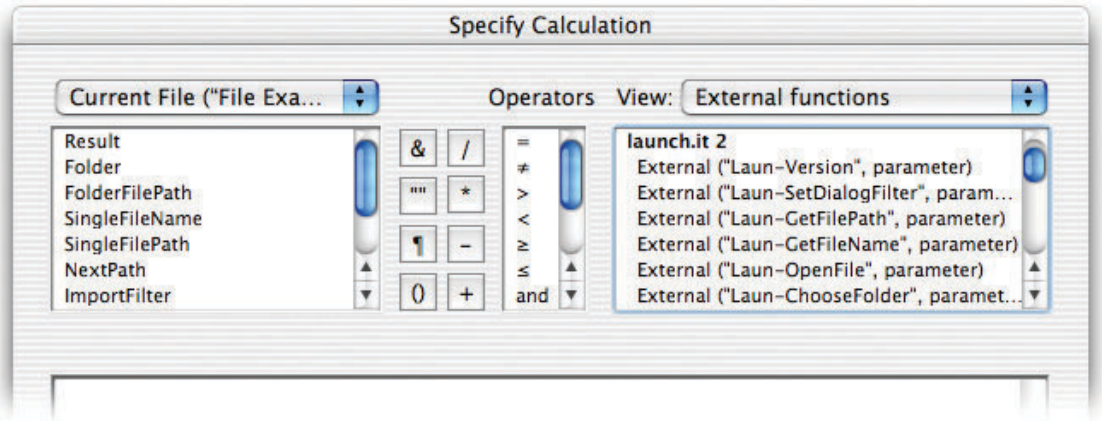
Viewing Imported Icons Cross-plattform - FileMaker limitation

When sharing a database in a cross-platform network, icons which have been imported under Mac OS X or Mac Classic can be viewed with FileMaker on Windows. However, if you import a file icon on Windows it can only be viewed with the Windows and the Mac Classic version of FileMaker. The Mac OS X version of FileMaker 5/6 can not display icons (or pictures) which have been imported on Windows using the BMP graphic format.

4. External functions

External functions are the interface for you as FileMaker developer to access the functionality of FileMaker plug-ins such as launch.it. This chapter will give you a detailed overview of all functions provided by launch.it 2 and their parameters.

Normally you make use of these functions as follows:
Create a global text field called *Result* in your FileMaker file which has the data type *Text*. This field will contain results returned by the plug-in. Next, create a new script using the *ScriptMaker* and add the script step *Set Field*. Specify the field *Result* as the target field for this command. Click the *Specify* button to open the FileMaker calculation editor. Now click on the *View* pop up menu and choose the section *External Functions*. You will find the mail.it functions in the section *launch.it 2*.



Not all functions are supposed to return their result to a global field such as a *Result* field. Some of launch.it's functions return content (like the path of a file) which are to be stored in separated FileMaker fields/records.

Function Name	**Laun-Version**
Syntax	External ("Laun-Version"; parameter)
Description	This function returns the version string of the plug-in that is currently installed. If launch.it is not installed an empty result is returned.
Parameter	No parameter needed, please use "".
Hint	Use this function in a start script to check whether the plug-in is installed. The FileMaker function 'TextToNum' lets you check whether a specific version (i.e. > 1) is active. Moreover, you can use this value in networks to perform updates with the AutoUpdate plug-in and FileMaker Server.

Function Name	**Laun-SetDialogFilter**
Syntax	External ("Laun-SetDialogFilter"; parameter)
Description	This function sets the file type filter for the File Open dialog which is shown by the function Laun-GetFilePath. The function returns "000 (OK)" if the filter could be set or "-001 (Incorrect option)" in case that an invalid filter has been passed to the plug-in.
Parameter	Use the parameter to set the file type filter for the File Open

dialog. A file type filter is formatted like *Document type;*.Extension* Use the ";" to separate several filters such as *Adobe Acrobat;*.pdf;Microsoft Word;*doc*. Use *.** for an *All Files* filter. Empty file filters will be interpreted like *.*

Function Name	**Laun-GetFilePath**
Syntax	External ("Laun-GetFilePath"; parameter)
Description	This function shows a File Open dialog and lets the user select a file. The full path of the file will be returned as result. If the user cancels the dialog an empty result will be returned.
Parameter	Use the parameter to set a title for the File Open dialog.

Function Name	**Laun-GetFileName**
Syntax	External ("Laun-GetFileName"; parameter)
Description	Returns the file name (including extension) of a certain file.
Parameter	The full file path has to be passed to the plug-in as parameter. Use Laun-GetFilePath or Laun-GetNextFilePath to retrieve full file paths.

Function Name	**Laun-Open File**
Syntax	External ("Laun-OpenFile"; parameter)
Description	Opens the specified file with the appropriate application. This function returns “000 (OK)” if the specified file could be opened. Otherwise an error code is returned.
Parameter	The full file path has to be passed to the plug-in as parameter. Use Laun-GetFilePath or Laun-GetNextFilePath to retrieve full file paths

Function Name	**Laun-ChooseFolder**
Syntax	External ("Laun-ChooseFolder"; parameter)
Description	Shows a Choose Folder dialog which lets the user choose any local or network folder. As result the full path of the selected folder is returned. If the user cancels the dialog an empty result is returned. This function may also return an error code.
Parameter	Use the parameter to set a custom title for the Choose Folder dialog.

Function Name	**Laun-UseSubfolders**
Syntax	External ("Laun-UseSubfolders"; parameter)
Description	Specifies whether subfolder are to be included when importing file paths from the folder set by Laun-SetFolder using Laun-GetNextFilePath. The function returns “000 (OK)” if it succeeded or an error code.
Parameter	Use “ON” as parameter to include subfolder or “OFF” if subfolders are not supposed to be included.
Hint	Call this function before you set a folder using Laun-SetFolder.

Function Name	**Laun-SetFileFilter**
Syntax	External ("Laun-SetFileFilter"; parameter)
Description	Sets the file type filter for the file paths to be imported form the folder specified by Laun-SetFolder. If the file type filter could be set “000 (OK)” is returned as result. In other cases an error code is returned.
Parameter	Use the parameter to pass the file type import filter to the plug-in. A file type is formatted like this: **.Extension*. Use the “;” to set a filter which imports file paths of several file types such as: **.pdf;*.doc*
Hint	Call this function before you set a folder using Laun-SetFolder.

Function Name	**Laun-SetFolder**
Syntax	External ("Laun-SetFolder"; parameter)
Description	Use the function Laun-SetFolder to display a *Choose Folder* dialog which lets you set a folder from which you would like to import file paths. The result returned is “000 (OK)” if the folder could be set. In any other case an error code will be returned.
Parameter	A full folder path is required as parameter. Use the function Laun-ChooseFolder to display a Choose Folder which lets the user select any local or network folder.
Hint	After setting a folder using this function you can use the function Laun-GetNextFilePath in a script loop to import all file paths from the specified folder.

Function Name	**Laun-GetNextFilePath**
Syntax	External ("Laun-GetNextFilePath"; parameter)
Description	This function returns the next file path from the folder specified by the parameter of the function Laun-SetFolder. The result returned by this function is the complete next file paths. If there is no further file in the specified folder an empty result is returned.
Parameter	No parameter needed, please use “”.
Hint	Use this function in a script loop to import all file paths from the folder specified by Laun-SetFolder. Note: When importing the file paths the plug-in behaves according to the settings specified by Laun-UseSubfolders, Laun-SetFileFilter, and Laun-SetFolder. Call this function after the parameters of those three have been set.

Function Name	**Laun-CopyLargeFileIcon**
Syntax	External ("Laun-CopyLargeFileIcon"; parameter)
Description	Use this function to copy a large (32x32 pixels) version of a file icon to the clipboard. “000 (OK)” is returned as result if the icon could be copied successfully. Otherwise an error code will be returned.
Parameter	The full path of the file is required. Use Laun-GetFilePath or Laun-GetNextFilePath to retrieve full file paths into FileMaker.
Hint	In order to paste the copied icon into a FileMaker container field use the script step *Paste[Container Field]*. Please note that the

container field has to be present in the current layout. Otherwise the Paste step will have no effect. In order to restore the clipboard after icon copy/paste processes the function Laun-RestoreClipboard can be used.

Function Name	**Laun-CopySmallFileIcon**
Syntax	External ("Laun-CopySmallFileIcon"; parameter)
Description	Use this function to copy a small (16x16 pixels) version of a file icon to the clipboard. “000 (OK)” is returned as result if the icon could be copied successfully. Otherwise an error code will be returned.
Parameter	The full path of the file is required. Use Laun-GetFilePath or Laun-GetNextFilePath to retrieve full file paths into FileMaker.
Hint	In order to paste the copied icon into a FileMaker container field use the script step *Paste[Container Field]*. Please note that the container field as to be present in the current layout. Otherwise the Paste step will have no effect. Use the small version of a file icon for list views in FileMaker. In order to restore the clipboard after icon copy/paste processes the function Laun-RestoreClipboard can be used.

Function Name	**Laun-RestoreClipboard**
Syntax	External ("Laun-RestoreClipboard"; parameter)
Description	This function restores the content of the clipboard which was saved automatically when calling the function Laun-CopyLargeFileIcon or Laun-CopySmallFileIcon.
Parameter	No parameter needed, please use “”.
Hint	On Windows only standard clipboard content such as plain text or bitmaps can be restored using this function

Function Name	**Laun-Set-Locale**
Syntax	External ("Laun-Set-Locale"; parameter)
Description	Sets the locale for the plug-in. This is needed to let the plug-in know how special characters such as French or German letters have to be converted. “000 (OK)” is returned as result.
Parameter	Use “English” to switch off the conversion of special characters. Use “Western” to show special characters such as French, German or Scandinavian ones correctly when importing file paths and names. Use “Cyrillic” to use the plug-in with Cyrillic letters. The default setting of this function is “Western”.
Hint	This function is available on Windows only. It is not needed by the Mac version of the plug-into interpret special characters in a path correctly.

Function Name	**Laun-Set-RegName**
Syntax	External ("Laun-Set-RegName"; parameter)
Description	Use this function to specify the registration name which will be provided by Dacons after you purchased a license of the plug-in. Set this value in a start script of your solution BEFORE Laun-Set-RegCode.
Parameter	The registration name provided by Dacons after you purchased a license.
Hint	Set the registration name value using this function BEFORE you use Laun-Set-RegName to specify the registration code.

Function Name	**Laun-Set-RegCode**
Syntax	External ("Laun-Set-RegCode"; parameter)
Description	This function sets the registration code which will be provided by Dacons after you purchased a license of the plug-in. Set this value in a start script of your solution AFTER Laun-Set-RegName. If the registration was successful this function returns “000 (OK)” which means that all trial restrictions will be removed for the current FileMaker session.
Parameter	The registration code provided by Dacons after you purchased a license.
Hint	You can also store the registration in the plug-in preferences if you do not like to register every session using external functions.

5. Error code handling

As mentioned in the External Functions chapter many of the launch.it functions return error codes if certain operations or commands fail. This allows you to implement an advanced error code handling into your FileMaker Pro solution. You can generate custom error messages in your language or LOG database entries.

Most of the functions return the code “000 (OK)” if no error occurs. The easiest way is to use the FileMaker text function *Left(Result; 3) = “000”* to check for this result.

The result codes which are returned if errors occur are all negative. In general they are formatted like this:

- Error Code (Description)

5.1 Error codes overview

The plug-in returns the following error codes:

000	(OK - no error)
- 001	(Incorrect option (incorrect parameter). This result can be returned if filters which have been set are not valid.)
- 002	(Directory not found. This can happen when importing files from a folder by setting the folder using the function Laun-SetFolder.)
- 003	(Cannot iterate directory. This might happen if the folder content changed during scan of directory content using Laun-SetFolder.)
- 004	(Might be returned by Laun-OpenFile if the file which is to be opened does not exist.)
- 005	(Error launching file. Might be returned by Laun-Open file if an error occurred while trying to open the specified file.)
- 006	(Cannot copy icon. This can be the case when calling the function Laun-CopyLargeFileIcon or Laun-CopySmalFileIcon.)
- 007	(Cannot restore clipboard content. Might be returned when calling the function Laun-RestoreClipboard. This result is also returned if the function Laun-RestoreClipboard is called before and icon has been copied.)

Check This Out!

Dacons is the creator of other great cross platform FileMaker Pro plug-ins.

mail.it

Main Features:
- Send/receive email, including attachments with FileMaker
- Incoming attachments can be stored in any local or network folder
- Cc, Bcc, Forward and Reply support
- Lightning fast sending and receiving
- Integrated email account manager
- Full HTML (rich content) support for incoming and outgoing emails
- Create HTML inline emails for powerful mass-mailings
- Check for email in a user specified interval - scriptable timer
- Check for mail from a specific sender or subject before you download
- Customizable status bar with a 'Cancel' button
- Priority settings (very high to very low)
- SMTP authentication supporting for all common protocols
- Leave messages on server after download for multiple use
- Request email read-receipt notification for outgoing messages
- Full email header access for incoming and outgoing emails
- Combine email and database updates with custom header fields
- Advanced error code handling
- FileMaker Runtime support
- Helps eliminate SPAM - keep your email traffic to an optimum
- Completely rewritten to accommodate user requests

Check out:

http://mailit.dacons.net

for a free trial version today!

MenuControl

MenuControl will enable you to create your own real menus in FileMaker Pro using any scripts as menu items and switch off the standard FileMaker menu! A powerful MenuComposer enables you to create your new menu in minutes. Here are some of MenuControl¹s powerful features:

Main Features:
- Create your own FileMaker Pro menu items
- Create your own submenus
- Create different menus for different layouts and files
- Create different menus for different users - switch menus at any time
- Added security for your FileMaker Pro solutions
- Full control over user navigation and access
- Create professional looking solutions with clean, uncluttered menus
- Assign any shortcuts to your scripts/menus
- Use more shortcuts and less buttons
- A simple MenuComposer makes creating new menus a breeze

Find out much more about MenuControl at:

http://menucontrol.dacons.net

and download a free, fully working trial version today!