



Version 1.0

The File Transfer Protocol Plug-in for Filemaker Pro 4.x / 5.x

Developed by Comm-Unity Networking Systems

Upload and Download
Files to an FTP site using
only FileMaker Pro!



FTPit Support Email:
FTPit@comm-unity.net

FTPit Support Website:
<http://FTPit.cnsplug-ins.com/>

Comm-Unity Networking Systems

Table of Contents

Introduction	3
Installation and Configuration	4
Basics	5
About	6
Registering	7
How to Use FileMaker Plug-ins	8
Working with FTPit	9
Text Upload Example Explained	10
Upload Script	11
FTPit External Function Reference	13
Understanding Error Responses	25
Credits	29
Contact Information	30



Introduction

FTPit is an exciting new plug-in brought to you by the makers of the leading email plug-ins, SMTPit and POP3it. FTPit uses the File Transfer Protocol (FTP) to allow you to upload and download files to and from an FTP site.

FTPit also allows you to upload text from a FileMaker Pro field as a text file on an FTP site. This can be used in a solution that creates static web pages for a web site. Instead of launching a separate FTP client to upload the html files your solution creates, you can do it all from within FileMaker Pro with the click of a button. Imagine the possibilities.

Features

- Upload or download any file you have access to on any FTP site.
- Upload a full static webpage using only FileMaker Pro and FTPit.
- Queue up multiple uploads and downloads and sit back while they all get uploaded or downloaded.
- Watch everything that FTPit is doing with its Status Window.
- Catalog all the files on an FTP site using FTPit's directory listing functions.

Installation and Configuration

To install the plug-in, first make sure FileMaker Pro is closed. Next, unzip the FTPit zip file on Windows, or unstuff the FTPit stuffit file on Macintosh, and then double-click the FTPit_Installer application. This will automatically place the FTPit plug-in file into the "System" folder on Windows, or the "FileMaker Extensions" folder on Macintosh, inside your FileMaker Pro 4.x or 5.x folder. If you have a previous version of FTPit, the installer will overwrite it.

(Windows Users: The Installer will not run from inside an Unzipping program like WinZip. You must unzip it to a folder before running the installer. Also, if you have WinZip and you have it set to automatically install applications that you download, you will not get the full benefit of this download. WinZip only installs the plug-in and does not show you the contents of the zip file, which contains the example databases and the documentation. You must unzip the FTPit zip file manually so that you can see the entire contents and not just the installer. If you set the WinZip "Wizard" interface to "Classic", it will not do this.)

After you install the plug-in as described above, you can open FileMaker Pro and set the default preferences. To do this, go to Edit->Preferences->Application, click on the Plug-ins tab, and double-click the FTPit plug-in.

Basics

Once the Configuration Dialog is open, click the **Basics** tab (See Figure 1) where you can set the following default values. **FTP Host** is where you enter the domain name or IP address of the FTP site you want to connect to. **Username** is where you enter the default username of the FTP account that you want to connect to. **Password** is where you enter the default password for the FTP account you want to connect to.

FTPit will refer to the default settings when you do not specifically set values in your scripts. In other words, if you do not define a username in your FTPit related script, FTPit will use the **Default Username**.

FTPit waits for certain amount of time for your FTP host to respond. By default, this is set to 15 seconds. However, you can adjust the setting to fit your needs by entering a different value in the **Connection Timeout** field.

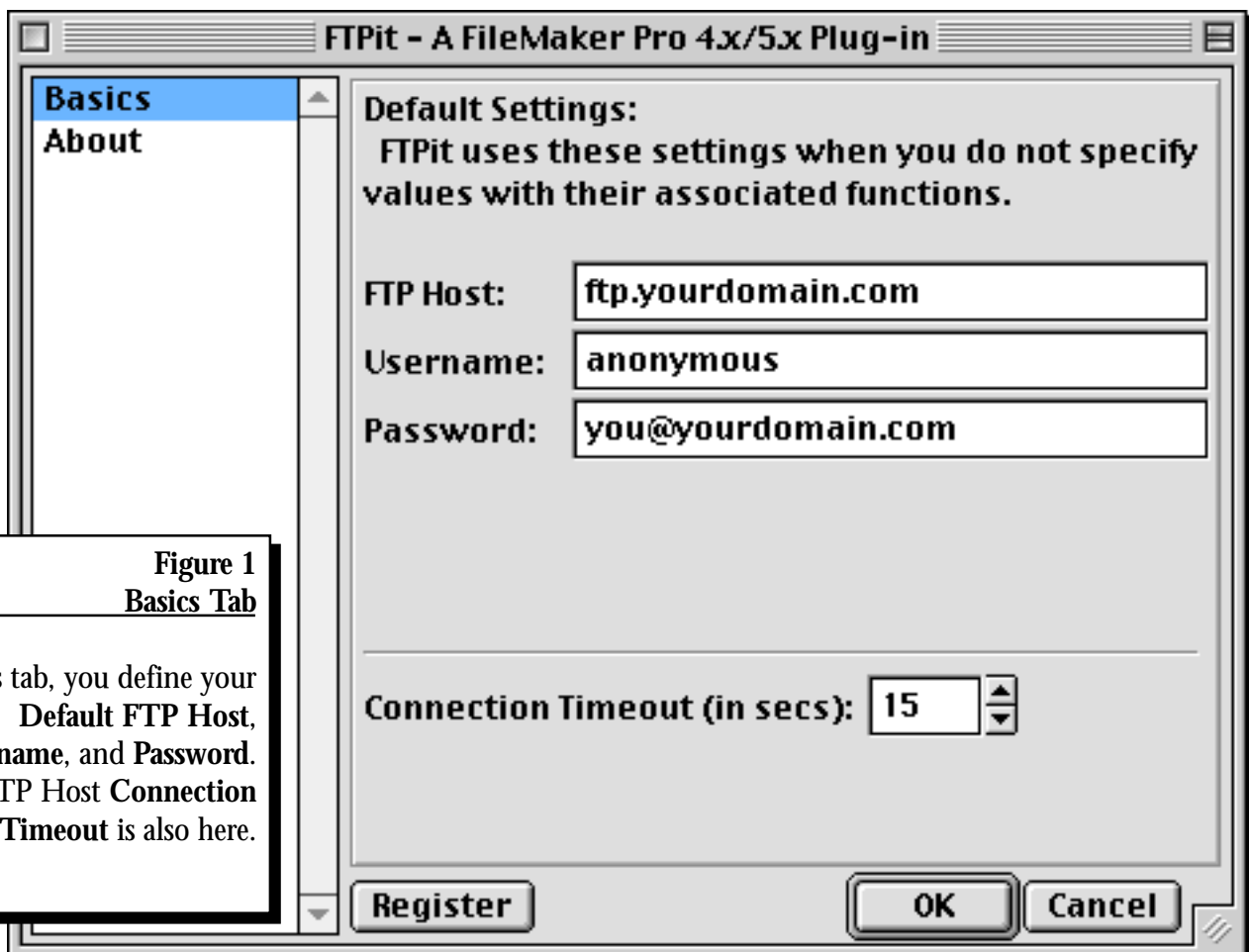


Figure 1
Basics Tab

On this tab, you define your
Default FTP Host,
Username, and Password.
The FTP Host Connection
Timeout is also here.

About

The **About** Tab (See Figure 2) simply tells you which version of FTPit you are currently running. It also displays who this copy of FTPit is registered to. If you do not have the most recent version that is listed on our website (<http://www.cnsplug-ins.com>), then you can upgrade simply by downloading and installing the newer version.

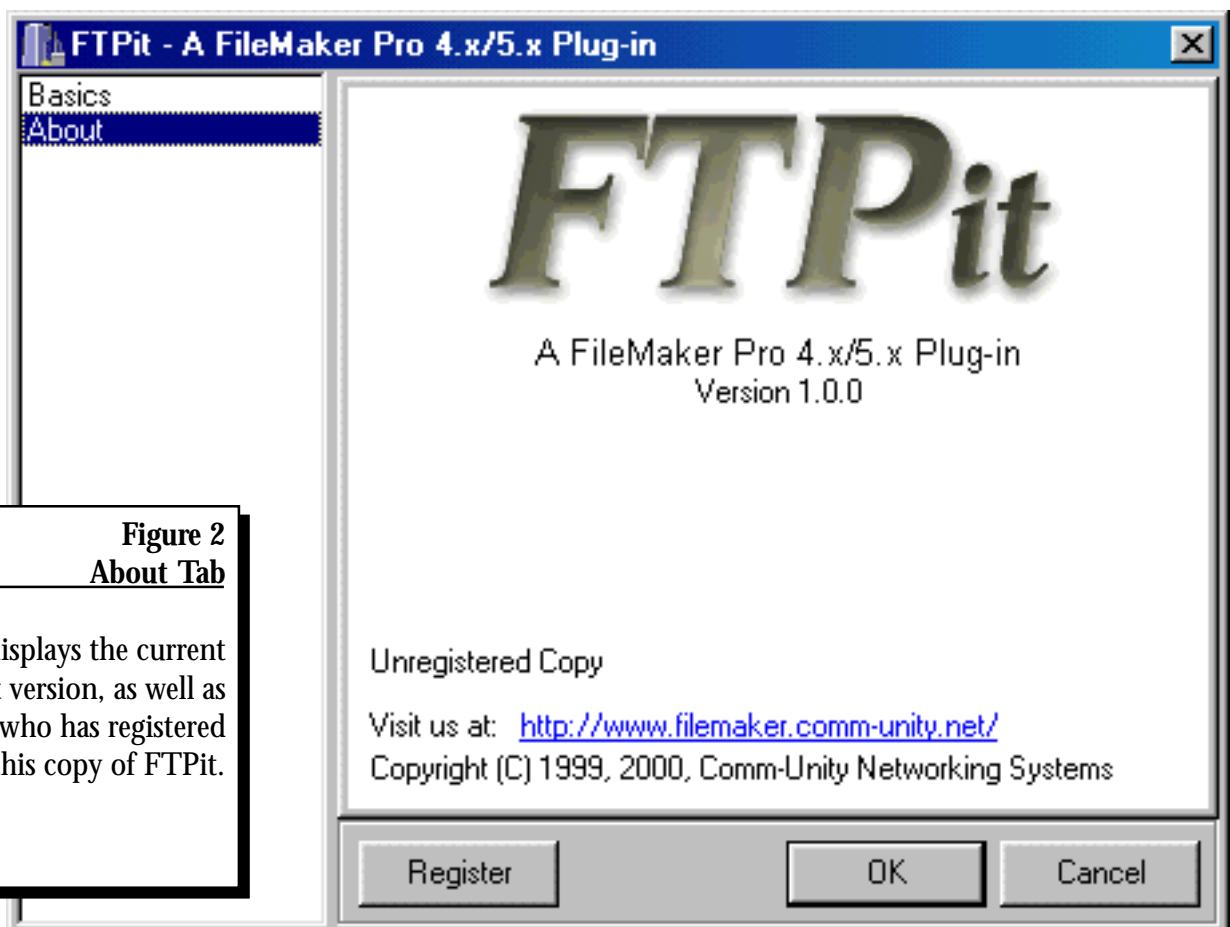


Figure 2
About Tab

This tab displays the current FTPit version, as well as displays who has registered this copy of FTPit.

Registering

You can register FTPit from the Configuration Dialog (See Figure 3) once you have purchased it from our secure website. After you purchase FTPit, we will send you a serial number to register your copy by using the Register button in the Configuration Dialog. We have also decided to include a **doFTP-Register** function so that developers can easily register FTPit with their (bound) solutions once they have purchased a Developer's License. For more information on purchasing FTPit and other exciting plug-ins, visit our website (<http://www.cnsplug-ins.com>) and choose the "Purchase" link from the tool bar.

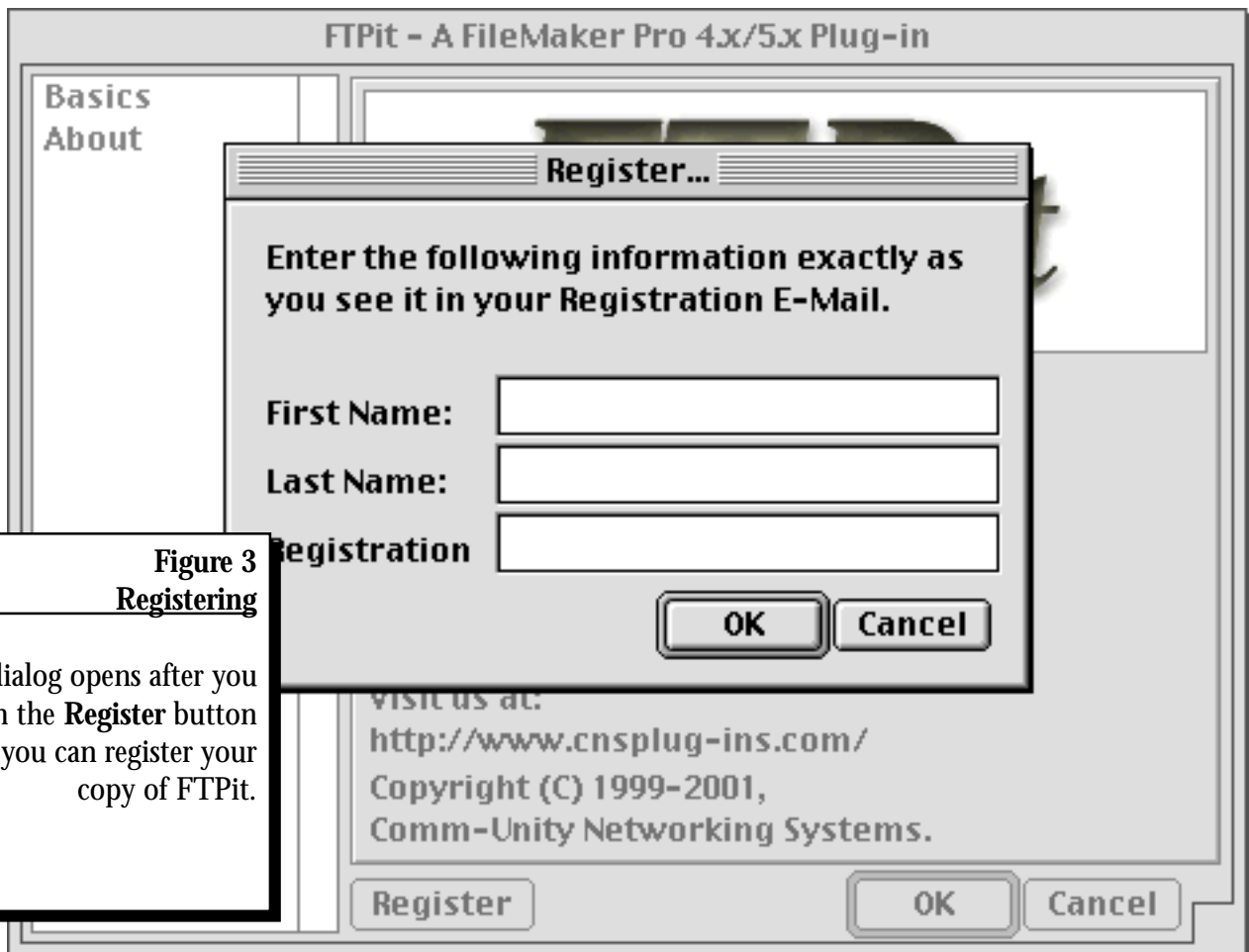


Figure 3
Registering

This dialog opens after you click on the **Register** button so that you can register your copy of FTPit.

How to Use FileMaker Pro Plug-ins

FileMaker, Inc. introduced a very simple plug-in architecture when they released FileMaker Pro 4.0. Originally intended to aid only in complex calculations, the plug-in architecture took off in ways that FileMaker, Inc. had not expected. Though there are now many different types of plug-ins, they all work the same basic way. By understanding the basics of plug-in interaction you will be able to understand how many developers approach plug-in development.

To start with, plug-ins are used by creating calculations. You can use calculations in many places within FileMaker Pro including a calculation field, the **Set Field** script step, the **Paste Result** script step, as well as a text field that has a calculated value. There are a couple of other ways to create calculations, but these are the major avenues that are currently being used for plug-ins. The most commonly used avenue for plug-ins is the **Set Field** script step. It is easy to use, plus, the plug-in can report its actions in the field that you are setting. If you have never used scripts before, you can find a complete explanation of scripts and script steps in the FileMaker Pro Help Topics located in the Help menu.

FileMaker's External Plug-in architecture is as follows:

External ("PluginNamePrefix-FunctionName", parameter)

For example if you take our **doFTP-Version** function, which returns the version of FTPit that you are using, it would look like this:

External ("doFTP-Version", parameter)

To actually use the function though, you will have to give it a parameter. Since the **doFTP-Version** function does not need any special information, you can just set it to the empty string ("") in order to use the function:

External ("doFTP-Version", "")

The parameter of the plug-in architecture is where you put the information for the related function. For example, if you are setting your Username using the **doFTP-Username** function, it would look like this:

External ("doFTP-Username", "frank")

Or if you are using a text field or a global field named "UsernameField" to hold your Username value, it would look like this:

External ("doFTP-Username", UsernameField)

The difference between using a field to hold the value and putting the real value into the calculation is the quotation marks. This is because FileMaker Pro will interpret anything not in quotes as a field in your database.

Though the calculation dialog box is limited in space, FileMaker Pro has made it somewhat easier to deal with functions by using their groupings of functions in the top right corner of the calculation dialog. To view all of the plug-in functions currently installed, choose view by "External Functions". Once chosen, you should see the WebCompanion external functions (if WebCompanion is installed and enabled), and if you have FTPit installed (and enabled) you will see the FTPit functions listed as well. Choosing to view by "External Functions" can greatly increase the ease of your script writing because the functions that you need are right there at your finger tips.

Working with FTPit

This section of the documentation explains a few fundamentals you need to understand when using FTPit to build your own custom solutions and to understand the example solutions provided for you.

The single most important concept is the idea of the "current" directory. All of the file related functions in FTPit use this concept of the "current" directory. Instead of referring to files by their full path (like "c:\My Documents\projects\my website\images\logo.gif"), you refer to them only by their filename (like "logo.gif"). This is accomplished by setting the "current" directory with the functions **doFTP-ChangeDir** and **doFTP-LocalChangeDir**. **doFTP-ChangeDir** changes the "current" remote directory (the directory of the FTP site you are connected to), while **doFTP-LocalChangeDir** changes the "current" local directory (the directory of your hard drive). Once you have set the "current" directory, you can use any of the file related functions on the files located in the "current" directory.

For example, let's say you wanted to upload the file "logo.gif" from the directory "c:\My Documents\projects\my website\images\" on your local hard drive to your ftp site in the directory "/users/frank/public_html/images/". The FTPit functions you would call are the following:

```
External("doFTP-LocalChangeDir", "c:\My Documents\projects\my website\images\") &  
External("doFTP-ChangeDir", "/users/frank/public_html/images/") &  
External("doFTP-Put", "logo.gif")
```

If there are other files in the same local directory that you want to upload, all you have to do is put in a few more function calls to **doFTP-Put**, like so:

```
External("doFTP-Put", "bannerad.gif")  
External("doFTP-Put", "product1.gif")
```

Another important concept is that FTPit uploads and downloads files in the "background", meaning that while FTPit is uploading or downloading files, you can continue to work with your database. We have included a status window in FTPit so that you know what FTPit is doing at all times. You can show, hide, and move this status window using the **doFTP-ShowStatus**, **doFTP-HideStatus**, and **doFTP-MoveStatus** functions explained in the **FTPit External Functions Reference** section later in this documentation.

You can also have FTPit call a script in your database when it is done uploading or downloading a file. You enable this by using the **doFTP-CompletedScript** function. FTPit will call the script that you define each time it has completed uploading or downloading a single file. So, if you have multiple "Puts" or "Gets", it will call the script for each one of the files you are "Putting" or "Getting". You can use the **doFTP-LastCompleted** function to determine which file was just completed and if you were "Putting" it or "Getting" it. You can use the **doFTP-GetQueue** function to see all the files that FTPit will be uploading or downloading, and to check to see if there are any more files being uploaded or downloaded. All of these functions are explained in the **FTPit External Functions Reference** section later in this documentation.

Text Upload Example Explained

This section of the documentation will show you, step by step, how the Text Upload Example database works so that you can understand how to use FTPit in your own solutions.

First of all there are seven fields defined in the solution that are used with FTPit. You will not necessarily need all of the fields mentioned in your FTP solution. The fields defined are the following:

Field Name	Field Type
FTPit Result	Text
Text Body	Text
FileName	Text
Host	Text
Username	Text
Password	Text
Path	Text

The **FTPit Result** field is used with the **Set Field** script steps. It is the field that is "set" with the results from the plug-in function calls. **Text Body** contains the "body" of the text file. You put everything in this field that you want to be in the final text file on the FTP site. **FileName** is the name of the file to create on the FTP site. **Host** contains the domain name or IP address of FTP site that you are connecting to. **Username** contains the username of the account on that FTP site. **Password** contains the password of the account on that FTP site. Finally, **Path** contains the path to the directory where you want this text file to be uploaded to.

Upload Script

Now that you know the fields in the database, let us look at the **Upload** script. When you view this script, you see that it is calling all of the other related scripts in order, and checking for errors after each step. Let us look at the first related script, **Set up**:

Script Step	Script Step Parameters
Set Field	Field Button: FTPit Result Specify Button: External("doFTP-Host", Host) & "¶" & External("doFTP-Username", Username) & "¶" & External("doFTP-Password", Password) & "¶" & External("doFTP-CompletedScript", "FTPit_Text_Upload_Example.FP3 Done")

The first three External calls set up the **Host**, **Username**, and **Password** using the fields in the database. The last External call tells FTPit to call the **Done** script in the FTPit_Text_Upload_Example.FP3 database when it has completed uploading the text.

The next script is the **Connect** script:

Script Step	Script Step Parameters
Set Field	Field Button: FTPit Result Specify Button: External("doFTP-Connect", "")

This simply calls the **doFTP-Connect** function to connect to the FTP site.

The next script is the **Change Directory** script:

Script Step	Script Step Parameters
Set Field	Field Button: FTPit Result Specify Button: External("doFTP-ChangeDir", Path)

This script calls the **doFTP-ChangeDir** function to change to the directory specified in the **Path** field in the database. This directory is the target directory for the file we will be uploading in the next step.

The next script is the **Put Text** script:

Script Step	Script Step Parameters
Set Field	Field Button: FTPit Result Specify Button: External("doFTP-TextAssign", Text Body)
Set Field	Field Button: FTPit Result Specify Button: External("doFTP-PutText", FileName)

The first **Set Field** script step assigns the "body" of the text file using the **Text Body** field in the database. This is how you assign the contents of the file that you will be uploading/creating on the FTP site. The second **Set Field** script step begins the text upload to your FTP site. The parameter for the **doFTP-PutText** function is the name of the file that you want FTPit to create on the FTP site.

The final scripts are the **Done** and **Disconnect** scripts. The **Disconnect** script is not called from the **Upload** script, but instead is called from the **Done** script. Since we do not want to disconnect from the FTP server before it has finished uploading the text, we have to wait until it is finished. This is why we set up a "CompletedScript" in the **Set up** script to call the **Done** script. When FTPit is finished, it will call the **Done** script, which simply calls the **Disconnect** script and tells the user it has completed uploading the text. The **Disconnect** script looks like this:

Script Step	Script Step Parameters
Set Field	Field Button: FTPit Result Specify Button: External("doFTP-Disconnect", "")

This simply calls the **doFTP-Disconnect** function to disconnect from the FTP site.

FTPit External Function Reference

The following is a list of all available functions and a complete description of each. Note that there are a few functions that can have default values set in the Configuration Dialog. See the Installation and Configuration section for a complete understanding of how the Configuration Dialog works.

doFTP-Version

This function returns the current version of FTPit when the parameter string is empty (""). You can also use this function to display the Configuration Dialog. If you pass it the parameter "CONFIGURE", the Configuration Dialog will open. If you pass it the parameter "ABOUT", it will display the Configuration Dialog with the **About** tab as the first tab shown.

Examples:

```
External("doFTP-Version", "")  
External("doFTP-Version", "CONFIGURE")  
External("doFTP-Version", "ABOUT")
```

doFTP-Register

This function allows you to register your copy of FTPit via a function rather than using the Configuration Dialog. It is mostly meant for developers so that they can register plug-ins for bound solutions. This function looks like the following:

```
External("doFTP-Register", "First Name|Last Name|Serial Number")
```

The parameter consists of your first name, last name, and serial number all separated by the pipe character ("|"). ("|" is created by typing shift-backslash.)

doFTP-Host

The Host is the domain name or IP address of the FTP site you are connecting to. You can assign the host in your scripts, or set up a Default Host in the Configuration Dialog. **doFTP-Connect** will return an error if no host has been set. If you assign a host with this function, FTPit will ignore the Default Host in the Configuration Dialog.

Example:

```
External("doFTP-Host", "ftp.filemaker.com")
```

See Also:

```
doFTP-Username  
doFTP-Password  
doFTP-Connect
```

doFTP-Username

Use the **doFTP-Username** function to assign the username of the FTP account you are connecting to. If you are connecting to an anonymous FTP site, use "anonymous".

Examples:

```
External("doFTP-Username", "frank")  
External("doFTP-Username", "anonymous")
```

See Also:

doFTP-Host
doFTP-Password
doFTP-Connect

doFTP-Password

Use the **doFTP-Password** function to assign the password of the FTP account you are connecting to. If you are connecting to an anonymous FTP site, use your email address.

Examples:

```
External("doFTP-Password", "love")  
External("doFTP-Password", "frank@domain.com")
```

See Also:

doFTP-Host
doFTP-Username
doFTP-Connect

doFTP-CompletedScript

Use this function to tell FTPit which script you would like it to call when it has completed uploading or downloading a file or if there was an error uploading or downloading a file. The parameter consists of the database the script is in, as well as the name of the script, separated by the pipe character ("|"). ("|" is created by typing shift-backslash.) You must specify the full name of the database, including the ".fp3" or ".fp5" extension. (Macintosh users may not have the extension, depending on how you originally named the file.)

Example:

```
External("doFTP-CompletedScript", "FTPit_Text_Upload_Example.fp3|Done")
```

See Also:

doFTP-LastCompleted

doFTP-LastCompleted

The **doFTP-LastCompleted** function tells you what file FTPit just completed uploading or downloading or the error that occurred when it tried to upload or download. You would use this function inside the script that the **doFTP-CompletedScript** calls. The parameter for this function should be the empty string (""). This function returns text in these forms:

```
GET: downloadedfilename.txt  
PUT: uploadedfilename.txt  
ERROR: GET: <error code>  
ERROR: PUT: <error code>
```

The first two are returned telling you that it successfully uploaded or downloaded a file. The second two are returned if there was an error uploading or downloading the file.

Example:

```
External("doFTP-LastCompleted", "")
```

See Also:

doFTP-CompletedScript

doFTP-Connect

This function opens a connection to the FTP site you defined as the Host and uses the Username and Password to log in. The parameter for this function should be the empty string ("").

Example:

```
External("doFTP-Connect", "")
```

See Also:

doFTP-Host
doFTP-Username
doFTP-Password
doFTP-Disconnect

doFTP-Disconnect

This function closes the connection to the FTP site. The parameter for this function should be the empty string ("").

Example:

```
External("doFTP-Disconnect", "")
```

See Also:

doFTP-Connect

doFTP-Type

This function sets the transfer type of the connection. The options are "BINARY" and "ASCII". Binary transfer type is the default and should always be used if you are downloading any sort of application or data file (like a database). You would use the ASCII transfer type when you are uploading or downloading a text file and you want the line endings to be converted to or from your local platform. Macintosh users have an extra option, "MACBINARY", which, if the FTP site supports it, will transfer both the data fork and resource fork of the file. (If you choose "MACBINARY" in the windows version, it will default back to "BINARY".)

Examples:

```
External("doFTP-Type", "ASCII")
External("doFTP-Type", "BINARY")
External("doFTP-Type", "MACBINARY") (Macintosh only)
```

doFTP-Passive

This function sets the connection type to Passive or Non-Passive for data transfers. By default this is Off, and you will most likely never have to change it. However, some FTP sites require Passive connections, and some Proxy servers require that you set it to On. So, if you are having trouble connecting to an FTP site, or you are using a Proxy server that requires it, call this function with a parameter of "On". Giving the parameter "Off" will turn Passive Mode back off.

Examples:

```
External("doFTP-Passive", "On")
External("doFTP-Passive", "Off")
```

doFTP-Get

This function retrieves a file from the current remote directory and saves it to the current local directory. You only specify the filename in the parameter, and not a full path. If the file is not in the current remote directory, use **doFTP-ChangeDir** first. Use **doFTP-LocalChangeDir** to specify where to save the file on the local hard drive.

If you call **doFTP-Get** with a new file while FTPit is uploading or downloading a separate file in the background, the new file will be added to a "queue" of files to be downloaded. When the current file has completed uploading or downloading, the new file will begin to download. You can use the **doFTP-GetQueue** function to return the current files that will be uploaded or downloaded.

Example:

```
External("doFTP-Get", "installer.zip")
```

See Also:

- doFTP-Type**
- doFTP-Passive**
- doFTP-Put**
- doFTP-PutText**
- doFTP-GetQueue**
- doFTP-ChangeDir**
- doFTP-LocalChangeDir**

doFTP-Put

This function takes a file from the current local directory and uploads it to the current remote directory. You only specify the filename in the parameter, and not a full path. If the file is not in the local directory, use **doFTP-LocalChangeDir** first. Use **doFTP-ChangeDir** to specify where to upload the file on the FTP site.

If you call **doFTP-Put** with a new file while FTPit is uploading or downloading a separate file in the background, the new file will be added to a "queue" of files to be uploaded. When the current file has completed uploading or downloading, the new file will begin to upload. You can use the **doFTP-GetQueue** function to return the current files that will be uploaded or downloaded.

Example:

```
External("doFTP-Put", "index.htm")
```

See Also:

- doFTP-Type**
- doFTP-Passive**
- doFTP-Get**
- doFTP-PutText**
- doFTP-GetQueue**
- doFTP-ChangeDir**
- doFTP-LocalChangeDir**

doFTP-TextAssign

Use this function when you are uploading text from a FileMaker Pro field to an FTP site (like in a custom static web page uploading database). This function clears out the current "Text" and assigns the parameter you gave. If you call this function twice, you will overwrite the text from the first call.

Examples:

```
External("doFTP-TextAssign", "<HTML><Body>This is the first part")
External("doFTP-TextAssign", AFieldInYourDatabase)
```

See Also:

doFTP-TextAppend
doFTP-PutText

doFTP-TextAppend

Use this function when you are uploading text from a FileMaker Pro field to an FTP site. This function adds the parameter you gave to the current "Text" that you assigned with **doFTP-TextAssign**. This helps you break the 64k limit of text imposed on FileMaker Pro fields if you have large amounts of text you need to upload. This can also be used in a record-looping script to add extra lines for each record. If you call **doFTP-TextAppend** and the current "Text" is empty, this will become the initial "Text" (in other words, you do not necessarily have to call **doFTP-TextAssign** the very first time).

Examples:

```
External("doFTP-TextAppend", "<li>" & ProductName & "</li>")
External("doFTP-TextAppend", AFieldInYourDatabase)
```

See Also:

doFTP-TextAssign
doFTP-PutText

doFTP-PutText

This function uploads the "Text" that you assigned with **doFTP-TextAssign** and **doFTP-TextAppend**. You specify the filename to upload it as in the parameter. Use **doFTP-ChangeDir** to specify where to upload the text on the FTP site.

If you call **doFTP-PutText** with some new text while FTPit is uploading or downloading a separate file or text in the background, the new text will be added to a "queue" of files to be uploaded. When the current file has completed uploading or downloading, the new text will begin to upload. You can use the **doFTP-GetQueue** function to return the current files that will be uploaded or downloaded.

Example:

```
External("doFTP-PutText", "products.htm")
```

See Also:

doFTP-Get
doFTP-Put
doFTP-TextAssign
doFTP-TextAppend
doFTP-GetQueue
doFTP-ChangeDir

doFTP-GetQueue

This function returns a list of the files you have queued for upload or download. You can queue multiple files for upload or download with multiple calls to **doFTP-Get**, **doFTP-Put**, and **doFTP-PutText**. The parameter should be the empty string ("").

You can also use this function to determine if FTPit is uploading or downloading anything. This function will return the empty string ("") when there is nothing on the queue, meaning nothing is uploading or downloading. It would be a good idea to check if there is anything uploading or downloading before you disconnect. If you disconnect while a file is being uploaded or downloaded, the file will only be partially uploaded or downloaded. Here is an example:

```
Set Field ["Queue", "External("doFTP-GetQueue", "")"]
If ["Queue <> """]
    Show Message ["You are currently uploading or downloading a file. Are you sure you
        want to disconnect?"]
    If ["Status(CurrentMessageChoice) = 1"]
        Exit Script
    End If
End If
Set Field ["FTPit Result", "External("doFTP-Disconnect", "")"]
```

Example:

```
External("doFTP-GetQueue", "")
```

See Also:

doFTP-Get
doFTP-Put
doFTP-PutText

doFTP-ChangeDir

This function changes the "current" remote directory to the path that you specify. This is the only function that takes a full path to a directory on the FTP site. All of the other functions work with files in the "current" remote directory that you set with this function. See the Working with FTPit section for a more complete understanding.

Examples:

```
External("doFTP-ChangeDir", "/pub/downloads/")  
External("doFTP-ChangeDir", "/users/frank/public_html/")
```

See Also:

doFTP-ChangeDirUp
doFTP-CurrentDir
doFTP-MakeDir
doFTP-RemoveDir

doFTP-ChangeDirUp

This function is a short cut to changing the current remote directory to its parent directory. For instance, if the current remote directory is "/pub/downloads/", after calling this function, the current remote directory would be "/pub/". The parameter should be the empty string ("").

Example:

```
External("doFTP-ChangeDirUp", "")
```

See Also:

doFTP-ChangeDir
doFTP-CurrentDir
doFTP-MakeDir
doFTP-RemoveDir

doFTP-CurrentDir

This function returns the current remote directory. This function is useful if the FTP site you log into automatically redirects you to your home directory. It is also useful if you are connected to a unix FTP site and you change directory to a "link". In general, it is a good idea to call **doFTP-CurrentDir** after calling **doFTP-ChangeDir** or **doFTP-ChangeDirUp** if you need to know exactly where you are on the FTP site at all times. The parameter should be the empty string ("").

Example:

```
External("doFTP-CurrentDir", "")
```

See Also:

doFTP-ChangeDir
doFTP-ChangeDirUp
doFTP-MakeDir
doFTP-RemoveDir

doFTP-MakeDir

Use this function to create a new directory in the current remote directory. Only specify the name of the new directory and not a full path. If you are creating a directory to put files in, remember to call

doFTP-ChangeDir to change the current remote directory to the new directory before calling **doFTP-Put** or **doFTP-PutText**.

Example:

```
External("doFTP-MakeDir", "new_dir")
```

See Also:

doFTP-ChangeDir
doFTP-ChangeDirUp
doFTP-CurrentDir
doFTP-RemoveDir

doFTP-RemoveDir

Use this function to delete a directory in the current remote directory. Only specify the name of the directory and not a full path. In general, FTP sites will not allow you to delete a directory if there are files in the directory; you would need to delete all the files in the directory before removing it.

Example:

```
External("doFTP-MakeDir", "old_backup")
```

See Also:

doFTP-ChangeDir
doFTP-ChangeDirUp
doFTP-CurrentDir
doFTP-MakeDir

doFTP-ListFirst

Use this function in conjunction with **doFTP-ListNext** to list all of the files in the current remote directory. **doFTP-ListFirst** will pull the complete file list from the FTP site, parse it into a useful list of files, and return the first file in the list. If there are no files to list, this function will return "ERROR: ListFirst: No Files." You should not call **doFTP-ListNext** if this function returns an error (doing so will do no harm, it just will not return any useful information). Refer to the **FTPit_Example** and **FTPit_Files** databases for an example on how to use **doFTP-ListFirst** and **doFTP-ListNext**.

Under normal circumstances, the parameter is an empty string (""). However, you can also specify a "wild card" to match against the filenames so that you only return a subset of files. For instance, if you only want to see the HTML files in the directory, you could specify a wild card like "*.htm". The wild cards are based on the wild cards you can use with the MSDos "Dir" command, if you are familiar with those. Here is a brief description:

"*" - Matches zero or more characters
"?" - Matches only one character
Any other character matches itself.

If you wanted to display all FileMaker Pro 4 and 5 files, you could specify the wild card "*.fp?". The "*" would match any filename, while the "fp?" would match "fp3" and "fp5". You should also note, that you may not always get what you want. For instances, the "*.fp?" wildcard could also match a file with the name "obscure.fpk".

Note: If you come across an FTP site where the files do not show up or do not appear to have all the information (missing filename, size, date, time), please contact us with the FTP site so that we can connect to it and correct the directory parsing routines to properly handle that FTP site.

Examples:

```
External("doFTP-ListFirst", "")  
External("doFTP-ListFirst", "*.fp?")
```

See Also:

doFTP-ListNext

doFTP-ListNext

Use this function to get the remaining FileNames in the current remote directory after calling **doFTP-ListFirst**. When this function returns the empty string (""), there are no more FileNames to retrieve. The parameter should be the empty string ("").

Example:

```
External("doFTP-ListNext", "")
```

See Also:

doFTP-ListFirst

doFTP-Size

Use this function to return the size of a file in the current remote directory. Only specify the filename and not a full path.

Example:

```
External("doFTP-Size", "installer.zip")
```

doFTP-Rename

Use this function to rename a file in the current remote directory. Only specify the filenames and not full paths. Specify the current name followed by the new name, separated with the colon (":"). For instance, if the file is currently named "index.htm" and you want to rename it to "index_backup.htm", use the parameter "index.htm:index_backup.htm". (Note: The colon character (":") is used to separate the files instead of the pipe character ("|"). The reason is because while a pipe

character can be used in a filename on macintosh and windows, a colon cannot. This allows you to rename a file with a pipe character in the name.)

Example:

```
External("doFTP-Rename", "index.htm:index_backup.htm")
```

doFTP-Delete

Use this function to delete a file in the current remote directory. Only specify the filename and not a full path.

Example:

```
External("doFTP-Delete", "index_backup.htm")
```

doFTP-GetLocalRoot

This function returns the "root" of the local hard drive. On windows, this function will always return "C:\". On macintosh, this function will return the name of the first "fixed" disk (your hard drive). The parameter should be the empty string ("").

Example:

```
External("doFTP-GetLocalRoot", "")
```

doFTP-LocalChangeDir

This function changes the "current" local directory to the path that you specify. This is the only function that takes a full path to a directory on your hard drive. All of the other functions work with files in the "current" local directory that you set with this function.

Examples:

```
External("doFTP-LocalChangeDir", "c:\myfiles\")
```

```
External("doFTP-LocalChangeDir", "Macintosh HD:myfiles:")
```

See Also:

doFTP-LocalChangeDirUp

doFTP-LocalCurrentDir

doFTP-LocalMakeDir

doFTP-LocalRemoveDir

doFTP-LocalChangeDirUp

This function is a short cut to changing the current local directory to its parent directory. For instance, if the current local directory is "c:\myfiles\downloads\" or "Macintosh HD:myfiles:downloads:", after calling this function, the current local directory would be "c:\myfiles\" or "Macintosh HD:myfiles:". The parameter should be the empty string ("").

Example:

```
External("doFTP-LocalChangeDirUp", "")
```

See Also:

doFTP-LocalChangeDir
doFTP-LocalCurrentDir
doFTP-LocalMakeDir
doFTP-LocalRemoveDir

doFTP-LocalCurrentDir

This function returns the current local directory. The parameter should be the empty string ("").

Example:

```
External("doFTP-LocalCurrentDir", "")
```

See Also:

doFTP-LocalChangeDir
doFTP-LocalChangeDirUp
doFTP-LocalMakeDir
doFTP-LocalRemoveDir

doFTP-LocalMakeDir

Use this function to create a new directory in the current local directory. Only specify the name of the new directory and not a full path. If you are creating a directory to download files to, remember to call **doFTP-LocalChangeDir** to change the current local directory to the new directory before calling **doFTP-Get**.

Example:

```
External("doFTP-LocalMakeDir", "new_dir")
```

See Also:

doFTP-LocalChangeDir
doFTP-LocalChangeDirUp
doFTP-LocalCurrentDir
doFTP-LocalRemoveDir

doFTP-LocalRemoveDir

Use this function to delete a directory in the current local directory. Only specify the name of the directory and not a full path. You cannot delete a directory if there are files in the directory; you would need to delete all the files in the directory before removing it.

Example:

```
External("doFTP-LocalMakeDir", "old_backup")
```

See Also:

```
doFTP-LocalChangeDir  
doFTP-LocalChangeDirUp  
doFTP-LocalCurrentDir  
doFTP-LocalMakeDir
```

doFTP-LocalListFirst

Use this function in conjunction with **doFTP-LocalListNext** to list all of the files in the current local directory. **doFTP-LocalListFirst** will pull the complete file list from your hard drive, parse it into a useful list of files, and return the first file in the list. If there are no files to list, this function will return "ERROR: LocalListFirst: No Files." You should not call **doFTP-LocalListNext** if this function returns an error (doing so will do no harm, it just will not return any useful information). Refer to the FTPit_Example and FTPit_Files databases for an example on how to use **doFTP-LocalListFirst** and **doFTP-LocalListNext**.

Under normal circumstances, the parameter is an empty string (""). However, you can also specify a "wild card" to match against the filenames so that you only return a subset of files. For an example of wild cards, see the **doFTP-ListFirst** function above.

Examples:

```
External("doFTP-LocalListFirst", "")  
External("doFTP-LocalListFirst", "*.fp?")
```

See Also:

```
doFTP-LocalListNext
```

doFTP-LocalListNext

Use this function to get the remaining FileNames in the current local directory after calling **doFTP-LocalListFirst**. When this function returns the empty string (""), there are no more FileNames to retrieve. The parameter should be the empty string ("").

Example:

```
External("doFTP-LocalListNext", "")
```

See Also:

```
doFTP-LocalListFirst
```

doFTP-LocalSize

Use this function to return the size of a file in the current local directory. Only specify the filename and not a full path.

Example:

```
External("doFTP-LocalSize", "installer.zip")
```

doFTP-LocalRename

Use this function to rename a file in the current local directory. Only specify the filenames and not full paths. Specify the current name followed by the new name, separated with the colon (":"). For instance, if the file is currently named "index.htm" and you want to rename it to "index_backup.htm", use the parameter "index.htm:index_backup.htm". (Note: The colon character (":") is used to separate the files instead of the pipe character ("|"). The reason is because while a pipe character can be used in a filename on macintosh and windows, a colon cannot. This allows you to rename a file with a pipe character in the name.)

Example:

```
External("doFTP-LocalRename", "index.htm:index_backup.htm")
```

doFTP-LocalDelete

Use this function to delete a file in the current local directory. Only specify the filename and not a full path.

Example:

```
External("doFTP-LocalDelete", "index_backup.htm")
```

doFTP-ShowStatus

Use this function to show a status window that displays information about what the plug-in is currently doing. For instance, when you are uploading or downloading a file, the status window will show you the file it is uploading or downloading as well as a progress bar indicating how much of the file has been uploaded or downloaded. If you specify an empty string ("") as the parameter, the status window will show up in the middle of the screen.

If you want to specify a starting location for the status window, specify the coordinates of the left and top of the dialog in pixels in the form "x,y" or "across,down". For instance, if you wanted to display it in the top right hand corner of your screen, and your screen resolution is set to 800x600, you could specify "700,100". If you specify a negative one ("-1") as either the x (across) or y (down) coordinates, the status window will be centered on that axis. For instance, if you want to display it on the bottom of your screen in the center, you would specify "-1,600"; or in the center of the screen by specifying "-1,-1".

On windows, this status window will always be on top of every other window and should never be hidden by another window. On macintosh, the status window can be hidden behind another open window. You can use the "Window" menu in the menu bar across the top of your screen to bring the status window to the front.

Also, on macintosh, you must ensure that the status window is closed before quitting the FileMaker Pro application. If the status window is still visible when you close the FileMaker Pro application, you will get an error message saying that FileMaker Pro cannot write to the disk, and give you the option to "Quit" or "Continue". You will only be able to "Quit", which is the equivalent of a force-quit, which does not safely free all its resources. The easiest way to ensure that the status window is closed when you close the FileMaker Pro application is to call **doFTP-HideStatus** in a "close" script that you define in Edit->Preferences->Document of your main database.

Examples:

```
External("doFTP-ShowStatus", "")  
External("doFTP-ShowStatus", "700,100")  
External("doFTP-ShowStatus", "-1,600")
```

See Also:

doFTP-HideStatus
doFTP-MoveStatus

doFTP-HideStatus

Use this function to hide the status window that you had previously shown with the **doFTP-ShowStatus** function. You must call this function to hide the status window before closing the FileMaker Pro application (see **doFTP-ShowStatus** above). Power-users will see that this function returns the last known coordinates of the status window, which can be extracted and stored so that the next time the status window is shown, it can be shown in the same place. The parameter should be the empty string ("").

Example:

```
External("doFTP-HideStatus", "")
```

See Also:

doFTP-ShowStatus
doFTP-MoveStatus

doFTP-MoveStatus

Use this function to move the status window to a different location on the screen. The parameter is the same as the parameter defined in the **doFTP-ShowStatus** function above. Power-users will see that this function returns the coordinates before and after the move, which can be extracted and stored so that the status window can be moved back to it's previous location if desired.

Examples:

```
External("doFTP-MoveStatus", "700,100")
```

```
External("doFTP-MoveStatus", "-1,600")
```

```
External("doFTP-MoveStatus", "-1,-1")
```

See Also:

doFTP-ShowStatus

doFTP-HideStatus

Understanding Error Responses

Every function in FTPit returns a response indicating the success or failure of that function. If the function is successful, it will return a response indicating that it set the value you were trying to set, completed the task that needed to be completed, or return the information that you requested. If however, the function is not successful, it will return an Error Response. This Error Response is in the form of:

ERROR: <Function Name>: <Error Description>

For example, if you forgot to set a Username using **doFTP-Username**, and you attempt to connect to the FTP site with **doFTP-Connect**, the **doFTP-Connect** function will return the following Error Response:

ERROR: Connect: Could not connect: No Username specified

Error responses always start with the word "ERROR" in all caps, followed by a colon, followed by the function that had the error, followed by a colon, followed by the actual error that occurred. You can use the various FileMaker Pro Text Functions to extract the different parts of the Error Response for your own use. For instance, if you want to know if the response you just got back was an ERROR, you can use the LeftWords function to return the first word and see if it is an error.

Example:

```
Set Field ["Result", "External("doFTP-Connect", "")"]
If ["LeftWords(Result, 1) = "ERROR"]
    <inform the user>
Else
    <upload a file>
End If
```

Credits

Programming, documentation, and example databases by Jake Traynham
Concept, web design, and example databases by Jesse Traynham

Contact Information

Email: FTPit@comm-unity.net
FTPit Website: <http://FTPit.cnsplug-ins.com/>
Main Website: <http://www.cnsplug-ins.com/>
Phone: 817-560-4226

You can write us at

Comm-Unity Networking Systems
8652 Hwy 80 West
Fort Worth, Texas 76116